Proceedings FOI-2015

Workshop on Foundations of Informatics

Institute of Mathematics and Computer Science
August 24-29, 2015, Chisinau, Moldova

CZU 004(082)

W 83

INSTITUTE OF MATHEMATICS AND COMPUTER SCIENCE
5, Academiei street, Chisinau, Republic of Moldova, MD 2028
Tel: (373 22) 72-59-82, Fax: (373 22) 73-80-27,
E-mail: imam@math.md
WEB address: http://www.math.md

Editors: Prof. S.Cojocaru, Prof. C.Gaindric.

Authors are fully responsible for the content of their papers.

Descrierea CIP a Camerei Naţionale a Cărţii

Workshop on Foundations of Informatics: Proceedings FOI-2015, August 24-29, 2015, Chişinău/Inst. of Mathematics and Computer Science, Acad. of Sciences of Moldova; ed.: S. Cojocaru, C. Gaindric. – Chişinău: Institut of Mathematics and Computer Science, 2015 (Tipogr. "Valinex SRL"). – 453 p.

Bibliogr. la sfârşitul art. – 100 ex.

ISBN 978-9975-4237-3-1.

004(082)

ISBN 978-9975-4237-3-1

This issue is supported by the Information Society Development Institute

Part 1

Theory of computing

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Structuring of Specification Modules*

(Invited paper)

Răzvan Diaconescu

Abstract

In this paper we first give a brief overview of the current status of modularisation for formal specifications and then we discuss a series of recent developments including parameter instantiation with sharing and module systems for behavioural specifications.

1 A Brief Introduction to Structuring of Specification Modules

The field of formal specification and verification of software and hardware systems is without alternative in safety-critical or security areas where one cannot take the risk of failure. Moreover formal specifications are crucial in ensuring a smooth development of large software systems as well as their evolution and maintainability. A special class of formal methods is given by the logic and algebraic based specifications languages. These support specification and verification methodologies that are rigorously backed up by formal logical systems, often display algebraic features that smoothen up significantly the verification process by integrating techniques with good computational properties such as rewriting. Modern algebraic specification systems include CASL [1], CafeOBJ [10], Maude [6], Specware [18], etc. Heterogeneous environments [10, 21] constitute a recent integrating trend in logic-based formal specification that provides a very flexible approach when choosing the

* This work has been supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-II-ID-PCE-2011-3-0439.

appropriate language and formalism. The theoretical infrastructure developed over many decades of research in formal specification has been exported also to other areas of computing science such as ontologies (e.g. [19]) or declarative programming.

Modularisation is the only way to cope with the complexity of formal specifications of large software systems. The work on specification modules has a long history that starts with the specification language Clear developed by Goguen and Burstall [5]. That laid general founding principles for module systems of formal specifications that have been realised by a multitude of subsequent languages and systems. However the current modularisation theory and methodologies include many subsequent developments that have been fuelled by the actual practice in formal specification and by the design of new modern languages and systems supporting this activity. Two quite major ideas have shaped the current approaches to specification modules. The first is to abstract away from the details of the logic underlying the actual specification language. The module composition techniques are largely independent of the logical formalism employed by the respective specification language; in fact this important observation constituted the main idea behind the language Clear and has triggered the conception of institution theory [14] as an abstract framework for modularisation. A second important idea envisaged in many works (e.g. [26, 2, 27], etc.) is to have a core set of specification building operators, with a clearly defined semantics, that can be used to define composition operators in actual module systems. These core specification building operators include *module sum* (or aggregation), *renaming* and *information hiding*. When the actual specification language provides tight semantics capabilities, then an initial or final semantics operator is also typically included. However occasionally (see [8]) these specification building operators do not suffice, therefore other operators have also to be considered. The recent work [7] develops modularisation theory in a way that is independent of any choice of specification building operators; for this another abstraction layer is introduced.

2 Genericity and Sharing

Generic or parameterised modules represent one of the most important module composition techniques because they can be (re)used in various ways by appropriate instantiations of their parameters. In simple terms, a parameterised module (denoted $\mathrm{SP}(P)$) can be regarded as a module import $P \to \mathrm{SP}$, with P being its *parameter* and SP being its *body*. Instantiation of parameterised modules is performed through interpretations of their parameters. In the specification literature they are usually called *views* and they are syntactic mappings $v\colon P \to \mathrm{SP}_1$ that satisfy two conditions:

1. they match consistently the signature of the parameter P to the signature of the value SP_1 (which is also a specification module); technically this amounts to the fact that v is a *signature morphism*; and
2. any implementation[1] of SP_1 has to be an implementation of the parameter P via the interpretation of the syntactic entities given by v.

Given a parameterised module $\mathrm{SP}(P)$ and a view $v\colon P \to \mathrm{SP}_1$ the *instance* $\mathrm{SP}(P \Leftarrow v)$ is commonly defined by the using the *pushout* technique (cf. [5, 27], etc.) from category theory [20], which informally can be explained as an extended form of union.

$$\begin{array}{ccc} P & \longrightarrow & \mathrm{SP} \\ {\scriptstyle v}\downarrow & & \downarrow \\ \mathrm{SP}_1 & \longrightarrow & \mathrm{SP}(P \Leftarrow v) \end{array} \qquad (1)$$

While this is the traditional approach to parameter instantiation which is widely employed by actual specification formalisms (e.g. [5, 1, 10, 6], etc.), it still raises several technical issues of important methodological significance and that can be summarised as follows:

[1] Mathematically speaking, 'implementations' are treated as models or interpretations in the underlying logic.

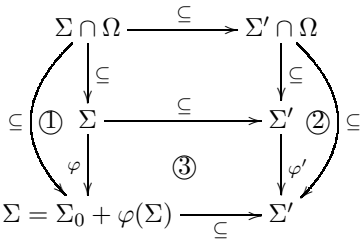
1. Since the pushout construction is unique only up to isomorphic renaming, actual implementations of module systems involving parameters provide ad-hoc constructions for the results of parameter instantiations. But how can we ensure in general that there exists an instantiation such that $\mathrm{SP}_1 \to \mathrm{SP}(P \Leftarrow v)$ behaves like an import, and moreover would this be uniquely defined?
2. In order to avoid technical complications the actual specification systems commonly dismiss the sharing between the body (SP) and the instance (SP_1), a situation that in practice constitutes a real restriction, as for example it may lead to duplication of the same data.
3. In the case of multiple parameters, for example $\mathrm{SP}(P_1, P_2)$, we can instantiate them sequentially (first $(\mathrm{SP}(P_1 \Leftarrow v_1)(P_2)$ and next $\mathrm{SP}(P_1 \Leftarrow v_1)(P_2 \Leftarrow v_2)$, or the other way around) or in parallel by regarding SP as parameterised by a single parameter, $\mathrm{SP}(P_1 + P_2 \Leftarrow v_1 + v_2)$. Are the two sequential instantiations and the parallel one equivalent methods in the sense of yielding isomorphic results?

The main technicalities involved in the answer to these questions include both a reshape of the definition of parameter instantiation and a general property of the signatures of the specification language formulated. In brief the traditional pushout square (1) has to be redefined as

$$\begin{array}{ccc} P \cup \mathrm{SP}_1 & \xrightarrow{\subseteq} & \mathrm{SP} \cup \mathrm{SP}_1 \\ {\scriptstyle v \cup 1_{\mathrm{SP}_1}}\downarrow & & \downarrow{\scriptstyle v'} \\ \mathrm{SP}_1 & \xrightarrow[i]{} & \mathrm{SP}(P \Leftarrow v) \end{array} \qquad (2)$$

(where $v \cup 1_{\mathrm{SP}_1}$ implies that v is identity on the part shared between P and SP_1)
and the signatures have to enjoy the following general property: for any signature morphism $\varphi\colon \Sigma \to \Sigma$ that is idempotent (i.e. such that $\varphi \circ \varphi = \varphi$) and such that there exists a signature Σ_0 such that Σ is the

disjoint union of Σ_0 and $\varphi(\Sigma)$



there exists a signature morphism $\varphi' \colon \Sigma' \to \Sigma'$ such that ③ is pushout and for any signature Ω such that ① commutes then ② commutes too.

In the works [8, 7, 28] the concepts of inclusion ($\subseteq$), union ($\cup$), intersection ($\cap$), disjointness, etc. are treated abstractly within the framework of 'inclusion systems' of [13]. In this way the solution proposed is general and can be applied to almost all existing specification formalisms (with the notable exception of behavioural specifications discussed below). The required property above holds naturally for all specification formalisms of interest, often in a stronger form than actually required (see [8, 28]). The generality of this solution implies also that it can be employed also by new specification formalisms to be defined in the future.

3 Module Systems for Behavioural Specifications

Modern algebraic specification theory and practice has extended the traditional many-sorted algebra-based specification to several new paradigms. One of the most promising is behavioural specification, which originates from the work of Horst Reichel [22, 23] and can be found in the literature under names such as hidden algebra [15, 16], observational logic [3, 17], coherent hidden algebra [11] and hidden logic [24]. Behavioural specification characterises how objects (and systems) *behave*, not how they are implemented. This new form of abstraction can be very powerful for the specification and verification

of software systems since it naturally embeds other useful paradigms such as concurrency, object-orientation, constraints, nondeterminism, etc. (see [16] for details). In the tradition of algebraic specification, the behavioural abstraction is achieved by using specification with hidden sorts and a behavioural concept of satisfaction based on the idea of indistinguishability of states that are observationally the same, which also generalizes process algebra and transition systems (see [16]). An important effort has been undertaken to develop languages and systems supporting the behavioural extension of conventional or less conventional algebraic specification techniques; these include CafeOBJ [10, 12], CIRC [25] and BOBJ [24]. In other situations, behavioural specification, although not directly realized at the level of the language definition, is employed as a mere methodological device [4]. In all cases there is the unavoidable need of a structuring mechanism for behavioural specifications.

However the modularisation of behavioural specifications poses specific challenges with respect to the standard modularisation techniques. Some important properties that in general are taken for granted do not hold in the case of behavioural specifications, for example the basic operation of union (aggregation) of behavioural specifications is only partial. The root cause of these problems lies in the 'encapsulation condition' on the signature morphisms, which prohibits new behavioural operations on old hidden sorts (i.e. that correspond to sorts of the source signature). Therefore the basic compositionality properties of behavioural specifications hold in a partial rather then total algebra style form. For example (see [9]) the associativity of union of behavioural specifications

$$(\mathrm{SP} \cup \mathrm{SP}') \cup \mathrm{SP}'' = \mathrm{SP} \cup (\mathrm{SP}' \cup \mathrm{SP}'')$$

means that either both members are defined and are equal semantically or else that neither of them is defined.

In the case of the instantiation of parameterised behavioural specifications the pushout square (1) is replaced with the following pushout

square:

$$\begin{array}{ccc} P \cup (\mathrm{SP} \cap \mathrm{SP}_1) & \xrightarrow{\subseteq} & \mathrm{SP} \\ {\scriptstyle v \cup \mathrm{id}} \downarrow & & \downarrow \\ \mathrm{SP}_1 & \longrightarrow & \mathrm{SP}'_1 \end{array} \tag{3}$$

when $P \cup \mathrm{SP}_1$ is defined. When $\mathrm{SP} \cup \mathrm{SP}_1$ is defined too, (3) can be replaced by the technically more convenient (2). However in this case the condition underlying (2) is stronger than that of (3) (note that according to [9] while unions of behavioural specifications are partial, intersections are total).

References

[1] Edigio Astesiano, Michel Bidoit, Hélène Kirchner, Berndt Krieg-Brückner, Peter Mosses, Don Sannella, and Andrzej Tarlecki. CASL: The common algebraic specification language. *Theoretical Computer Science*, 286(2):153–196, 2002.

[2] Jan Bergstra, Jan Heering, and Paul Klint. Module algebra. *Journal of the Association for Computing Machinery*, 37(2):335–372, 1990.

[3] Michel Bidoit, Rolf Hennicker, and Martin Wirsing. Behavioural and abstractor specifications. *Sci. Comput. Program.*, 25(2-3):149–186, 1995.

[4] Michel Bidoit, Donald Sannella, and Andrzej Tarlecki. Observational interpretation of CASL specifications. *Mathematical Structures in Computer Science*, 18(2):325–371, 2008.

[5] Rod Burstall and Joseph Goguen. The semantics of Clear, a specification language. In Dines Bjorner, editor, *1979 Copenhagen Winter School on Abstract Software Specification*, volume 86 of *Lecture Notes in Computer Science*, pages 292–332. Springer, 1980.

[6] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All About Maude - A High-Performance Logical Framework*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.

[7] Răzvan Diaconescu. An axiomatic approach to structuring specifications. *Theoretical Computer Science*, 433:20–42, 2012.

[8] Răzvan Diaconescu and Ionuţ Ţuţu. On the algebra of structured specifications. *Theoretical Computer Science*, 412(28):3145–3174, 2011.

[9] Răzvan Diaconescu and Ionuţ Ţuţu. Foundations for structuring behavioural specifications. *Journal of Logical and Algebraic Methods in Programming*, 83(3–4):319–338, 2014.

[10] Răzvan Diaconescu and Kokichi Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*, volume 6 of *AMAST Series in Computing*. World Scientific, 1998.

[11] Răzvan Diaconescu and Kokichi Futatsugi. Behavioural coherence in object-oriented algebraic specification. *Universal Computer Science*, 6(1):74–96, 2000. First version appeared as JAIST Technical Report IS-RR-98-0017F, June 1998.

[12] Răzvan Diaconescu and Kokichi Futatsugi. Logical foundations of CafeOBJ. *Theoretical Computer Science*, 285:289–318, 2002.

[13] Răzvan Diaconescu, Joseph Goguen, and Petros Stefaneas. Logical support for modularisation. In Gerard Huet and Gordon Plotkin, editors, *Logical Environments*, pages 83–130. Cambridge, 1993. Proceedings of a Workshop held in Edinburgh, Scotland, May 1991.

[14] Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, 1992.

[15] Joseph Goguen and Răzvan Diaconescu. Towards an algebraic semantics for the object paradigm. In Hartmut Ehrig and Fernando Orejas, editors, *Recent Trends in Data Type Specification*, volume 785 of *Lecture Notes in Computer Science*, pages 1–34. Springer, 1994.

[16] Joseph Goguen and Grant Malcolm. A hidden agenda. *Theoretical Computer Science*, 245(1):55–101, 2000.

[17] Rolf Hennicker and Michel Bidoit. Observational logic. In A. M. Haeberer, editor, *Algebraic Methodology and Software Technology*, number 1584 in LNCS, pages 263–277. Springer, 1999. Proc. AMAST'99.

[18] Kestrel Institute. *Specware system and documentation*, 2003. `www.specware.org`.

[19] Oliver Kutz, Till Mossakowski, and Dominik Lücke. Carnap, Goguen, and the hyperontologies - logical pluralism and heterogeneous structuring in ontology design. *Logica Universalis*, 4(2):255–333, 2010.

[20] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, second edition, 1998.

[21] T. Mossakowski, C. Maeder, and K. Lütich. The heterogeneous tool set. In *Lecture Notes in Computer Science*, volume 4424, pages 519–522. 2007.

[22] Horst Reichel. Behavioural equivalence – a unifying concept for initial and final specifications. In *Proceedings, Third Hungarian Computer Science Conference*. Akademiai Kiado, 1981. Budapest.

[23] Horst Reichel. *Initial Computability, Algebraic Specifications, and Partial Algebras*. Clarendon, 1987.

[24] Grigore Roşu. *Hidden Logic*. PhD thesis, University of California at San Diego, 2000.

[25] Grigore Roşu and Dorel Lucanu. Circular coinduction: A proof theoretical foundation. In Alexander Kurz, Marina Lenisa, and Andrzej Tarlecki, editors, *Algebra and Coalgebra in Computer Science*, volume 5728 of *Lecture Notes in Computer Science*, pages 127–144, 2009.

[26] Donald Sannella and Andrzej Tarlecki. Specifications in an arbitrary institution. *Information and Control*, 76:165–210, 1988.

[27] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specifications and Formal Software Development*. Springer, 2012.

[28] Ionuţ Ţuţu. Parameterisation for abstract structured specifications. *Theoretical Computer Science*, 517:102–142, 2014.

Răzvan Diaconescu

Received July 6, 2015

Simion Stoilow Institute of Mathematics of Romanian Academy
Romania
E–mail: Razvan.Diaconescu@imar.ro

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Small P Systems with Catalysts or Anti-Matter Simulating Generalized Register Machines and Generalized Counter Automata

Artiom Alhazov Rudolf Freund Petr Sosík

Abstract

In this paper we focus on two weak forms of cooperation in P systems, namely, catalytic rules and matter/anti-matter annihilation rules. These variants of P systems both are computationally complete, while the corresponding rule complexity turns out to be of special interest. For establishing considerably small universal P systems in both cases, we found two suitable tools: generalized register machines and generalized counter automata. Depending on the features used in the different variants, we construct several small universal P systems.

1 Introduction

Membrane systems with symbol objects are a theoretical framework of parallel distributed multiset processing, for example, see [12, 13, 14]. While non-cooperative P systems are known to characterize the regular languages, in case of unrestricted (even binary) cooperation, showing computational completeness is straightforward, for example, by simulating register machines. Hence, since many years researchers have been interested in even weaker forms of cooperation.

A catalytic rule is a non-cooperative rule with an additional catalyst on both the left side and the right side of the rule. Essentially, a catalyst only inhibits parallelism of rules where it is indicated. The question whether catalytic P systems are computationally complete (without priorities or other additional features) has been open for a number of

years, being finally answered positively, moreover, even showing that two catalysts suffice (or three for the purely catalytic systems), see [7].

In the variant with anti-matter objects, in addition to non-cooperative rules, specific cooperative erasing is allowed, namely, of two objects related by a bijection "object-antiobject". Anti-matter in P systems is a rather recent direction, for instance, see [1].

Small universal P systems have been investigated for a number of years. The smallest ones are those with string objects and splicing rules where even five rules suffice, see [5]. In the case of symbol objects, if full cooperation is allowed, then 23 rules suffice, see [6], and only 16 are needed if in addition inhibitors are allowed, see [8].

In this paper, we give an overview on small universal P systems using anti-matter or catalysts as in [4] and we even improve the results established there for (purely) catalytic P systems, based on recent results obtained in [3] as well as in [16] and [17].

2 Definitions

We assume the reader to be familiar with the basic notions and concepts from formal language theory, for example, see textbooks as [15]; for the area of P systems we refer to [12, 13, 14] and to [18] for actual news.

2.1 Register Machines

Register machines are well-known universal devices for computing (generating or accepting) sets of (vectors of) natural numbers.

Definition 1 *A* register machine *is a construct* $M = (m, B, l_0, l_h, P)$ *where*

- m *is the number of registers,*
- P *is the set of instructions bijectively labeled by elements of* B*,*
- $l_0 \in B$ *is the initial label, and*
- $l_h \in B$ *is the final label.*

The instructions of M can be of the following forms:

- $p : (ADD(r), q, s)$, *with* $p \in B \setminus \{l_h\}$, $q, s \in B$, $1 \leq r \leq m$.
  Increase the value of register r by one, and non-deterministically jump to instruction q or s.

- $p : (SUB(r), q, s)$, *with* $p \in B \setminus \{l_h\}$, $q, s \in B$, $1 \leq r \leq m$.
  *If the value of register r is not zero, then decrease the value of register r by one (*decrement *case) and jump to instruction q, otherwise jump to instruction s (*zero-test *case).*

- $l_h : HALT$.
  Stop the execution of the register machine.

A configuration *of a register machine is described by the contents of each register and by the value of the current label, which indicates the next instruction to be executed. M is called* deterministic *if all the ADD-instructions are of the form $p : (ADD(r), q)$.*

In the accepting case, a computation starts with the input of a k-vector of natural numbers in its first k registers and by executing the first instruction of P (labeled with l_0); it terminates with reaching the $HALT$-instruction. Without loss of generality, we may assume all registers to be empty at the end of the computation.

A register machine M_U is called *universal*, if, given the code of an arbitrary register machine M, M_U can simulate the computations of M on any given input. We speak of *strong universality*, if both input and output are given directly as numbers where as *weak universality* means that both input and output are encoded by a recursive function f, e.g., $f(n) = 2^n$; we also consider *weak-strong universality* with encoded input, but unencoded output.

2.2 P Systems

In this paper, we will only consider membrane systems with the simplest membrane structure $\mu = [\,]_1$, i.e., with even omitting μ, we consider a (catalytic) *P system* as a construct $\Pi = (O, C, w_1, R_1)$ where O

is the alphabet of *objects*, $C \subseteq O$ is the set of *catalysts*, w_1 the multiset of objects present in the skin region at the beginning of a computation, and R_1 is a finite set of *evolution rules*, associated with the skin region. In this paper we only use the *maximally parallel derivation mode*, i.e., in each derivation step we apply a non-extendable multiset of rules.

If a rule $u \to v$ has at least two objects in u, then it is called *cooperative*, otherwise it is called *non-cooperative*. In *catalytic P systems* we use non-cooperative as well as *catalytic rules*, which are of the form $ca \to cv$ where c is a special object called *catalyst*, which never evolves (this restriction can be relaxed), but it just assists object a to evolve to the multiset v. In a *purely catalytic P system* we only allow catalytic rules. If we allow catalysts to switch between different states, we speak of *multi-stable catalysts*.

In P systems with *anti-matter* objects, each object a also has an anti-matter object $\bar{a}$ in O and, in addition to non-cooperative and catalytic rules, matter/anti-matter annihilation rules $a\bar{a} \to \lambda$ are allowed, for instance, see [1].

In P systems with *toxic* objects, specific symbols are specified as being toxic; a computation can only be continued by a non-extendable multiset of rules which does not leave any toxic object idle. For more details about toxic P systems, for example, see [2].

3 Small Universal Register Machines

The universal register machines with the smallest known number of instructions are those constructed by I. Korec in [10]. For the standard instruction set (`ADD`-instructions and `SUB`-instructions, not counting the halting one), these are the strongly universal machine U_{22} and the weakly universal machine U_{20}, see Figure 1.

3.1 Generalized Register Machines

We often observe that the most efficient (in terms of rule complexity) simulations of register machines by P systems do not use separate rules for `ADD`-instructions, but perform them as a part of the rules simulating

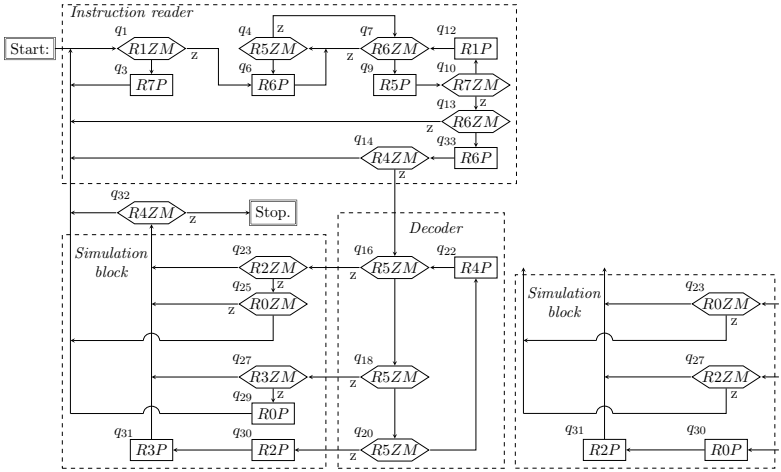


Figure 1. The strongly universal register machine U_{22} (left) and the simulation block of the weakly universal register machine U_{20} (right).

SUB-instructions. Hence, we recall from [4] the following generalization of register machines, as a tool for such simulations.

The model of *generalized register machines* has only instructions of one type except the halt instruction, i.e., generalized SUB-instructions of the form $j : (\mathtt{SUB}(r), A_-(j)k, A_0(j)l)$ where $j, k, l \in B$ are instruction labels and $A_-(j), A_0(j)$ are (possibly empty) strings of increment commands (sub-instructions) $\mathtt{ADD}(j')$. Clearly, a standard register machine (with ADD-instructions and SUB-instructions) can be obtained from a generalized one, simply by introducing intermediate states, see [4] for additional remarks.

3.2 With Multiple Registers

Below we present the (rules for the) strongly universal register machine U_{22} of Korec, see [10] and Figure 1, left, in the form of a generalized register machine:

$q_1 : (\texttt{SUB}(1), \texttt{ADD}(7)q_1, \texttt{ADD}(6)q_4)$, $q_{16} : (\texttt{SUB}(5), q_{18}, q_{23})$,
$q_4 : (\texttt{SUB}(5), \texttt{ADD}(6)q_4, q_7)$, $q_{18} : (\texttt{SUB}(5), q_{20}, q_{27})$,
$q_7 : (\texttt{SUB}(6), \texttt{ADD}(5)q_{10}, q_4)$, $q_{23} : (\texttt{SUB}(2), q_{32}, q_{25})$,
$q_{10} : (\texttt{SUB}(7), \texttt{ADD}(1)q_7, q_{13})$, $q_{25} : (\texttt{SUB}(0), q_1, q_{32})$,
$q_{13} : (\texttt{SUB}(6), \texttt{ADD}(6)q_{14}, q_1)$, $q_{27} : (\texttt{SUB}(3), q_{32}, \texttt{ADD}(0)q_1)$,
$q_{14} : (\texttt{SUB}(4), q_1, q_{16})$, $q_{32} : (\texttt{SUB}(4), q_1, q_h)$,
$q_{20} : (\texttt{SUB}(5), \texttt{ADD}(4)q_{16}, \texttt{ADD}(2)\texttt{ADD}(3)q_{32})$.

In the generalized register machine form of the weakly universal register machine U_{20} of Korec, see [10], q_{25} is no longer present, and instructions q_{20}, q_{23} and q_{27} are different, see Figure 1, right, and register 3 is not needed any more:

$q_{20} : (\texttt{SUB}(5), \texttt{ADD}(4)q_{16})$, $q_{23} : (\texttt{SUB}(0), q_{32}, q_1)$, $q_{27} : (\texttt{SUB}(2), q_{32}, q_1)$.

Remark 1 *Sometimes, also for technical reasons, we want to produce the output in a register which only has increment instructions associated to it, and have all other registers empty in the end. Unfortunately, these technical details are not fulfilled by the (strongly or weakly) universal register machines constructed by Korec in [10]: the result is obtained in register* 0*, a register allowing for* SUB*-instructions, and, due to the specific features of the register machines simulated by the universal Korec machines, (only) the registers* 1 *and* 6 *are not empty. Therefore, the last instruction* q_{32} *can be replaced by the following ones, with register* 8 *being the new output register; we can omit the right column and already take* q_{35} *as the halting state if "cleaning" is not needed:*

$q_{32} : (\texttt{SUB}(4), q_1, q_{34})$, $q_{35} : (\texttt{SUB}(1), q_{35}, q_{36})$,
$q_{34} : (\texttt{SUB}(0), \texttt{ADD}(8)q_{34}, q_{35})$, $q_{36} : (\texttt{SUB}(6), q_{36}, q_h)$.

3.3 With Two Decrementable Registers

In this subsection we discuss how to reduce the number of registers to two, possibly not counting an extra increment-only register. It is well known, e.g., see [11], that the computations of any m-register machine

can be simulated by a 2-register machine, via exponential encoding. Indeed, if we take the first m prime numbers p_i, $1 \leq i \leq m$, the values x_i of the registers i, $1 \leq i \leq m$, can be encoded in any of the first two registers as the single number ${p_1}^{x_1} \dots {p_m}^{x_m}$. Then, $\texttt{ADD}(r)$ is simulated by multiplying the value of the first register by p_r, and $\texttt{SUB}(r)$ is simulated by trying to divide the value of the first register by p_r; if the division is successful, the decrement transition is made, and otherwise, the value is restored, and the zero-test transition is made.

In the following, we analyze the simulation blocks mentioned above and represent the obtained 2-register machine in the generalized register machine form, following the constructions given in [11]:
– Instruction $j : (\texttt{ADD}(r), k)$ is simulated by the two generalized SUB-instructions $j : (\texttt{SUB}(1), (\texttt{ADD}(2))^{p_r}\ j, j')$ and $j' : (\texttt{SUB}(2), \texttt{ADD}(1)j', k)$.
– Instruction $j : (\texttt{SUB}(r), k, l)$ is simulated by the $p_r + 2$ generalized SUB-instructions

$$
\begin{aligned}
&j : (\texttt{SUB}(1), j_1, j'),\\
&j_n : (\texttt{SUB}(1), j_{n+1}, (\texttt{ADD}(1))^n\ j''), \text{ for } 1 \leq n \leq p_r - 2,\\
&j_n : (\texttt{SUB}(1), \texttt{ADD}(2)j, (\texttt{ADD}(1))^n\ j''), \text{ for } n = p_r - 1,\\
&j' : (\texttt{SUB}(2), \texttt{ADD}(1)j', k),\\
&j'' : (\texttt{SUB}(2), (\texttt{ADD}(1))^n\ j'', l), \text{ for } n = p_r.
\end{aligned}
$$

In the course of the analysis of the number of uses of decrements, the assignment of prime numbers to registers was chosen for U_{22}: The conditional decrement of register 5 happens 4 times, the conditional decrements of registers 4 and 6 happen twice each, and the conditional decrement of any other register happens once. Register 5 is represented by powers of 2, registers 6 and 4 by powers of 3 and 5 as well as registers 0, 1, 2, 7, and 3 by powers of 7, 11, 13, 17, and 19, respectively. We remark that we use a smaller prime for $R6$ than for $R4$ because the former is incremented twice and the latter is incremented only once in the underlying Korec machine, which, compared to the opposite choice leads to saving two ADD-instructions, which might be an interesting feature in another context, although in the present paper we are not concerned about that. We used the largest of the first 8 primes for $R3$ because $R3$, besides being one of the least-used registers here, is

no longer used in the weakly universal Korec machine U_{20} considered next. Moreover, a smaller prime is used for $R0$ than for $R1$ and $R2$, because $R0$ is also involved in the decoding phase discussed below.

In [4] the rule complexity of this reduction was improved as follows. It was noted that the recopying for increment and zero-test usually can be avoided by assigning different "master registers" to different states. The word "usually" means whenever the master register is changed after increment and zero-test, but is not changed after a decrement.

Hence, now the following allocation of master registers is chosen:

Register 1: q_1, q_6, q_9, q_{12}, q_{33}, q_{18}, q_{22}, q_{27}.

Register 2: q_3, q_4, q_7, q_{10}, q_{13}, q_{14}, q_{16}, q_{20}, q_{23}, q_{25}, q_{32}, q_h.

Another observation that we use to save even more instructions is the following: if an increment instruction has a unique entry point which is a zero-test, then such an increment can be embedded into the zero-test without using additional instructions. Clearly, the same transformation can be applied to multiple consecutive increments. The states q_{29}, q_{30}, and q_{31} do not appear in the register allocations above because we have embedded them into the zero-tests of q_{27} and q_{20}.

We proceed with evaluating the instruction complexity of the obtained generalized register machine by states. With the register allocation given above, recopying has been skipped for all transitions except $q_{13} \to q_1$ and $q_{23} \to q_{32}$. The table below shows the numbers of generalized register machine instructions associated with each generalized register machine instruction, and the numbers of generalized register machine instructions associated with the states q_{13} and q_{23} are underlined. The necessity of at least two recopyings can be argued by inspecting the cycles $q_1 - q_6 - q_4 - q_7 - q_9 - q_{10} - q_{13} - q_1$ and $q_{23} - q_{25} - q_{32} - q_{23}$; these cycles do not have common nodes, and each cycle needs at least one recopying to have the value in the original register. This minimality has been further confirmed by computer search in the space of possible allocations of registers to states, furthermore showing the uniqueness of the optimal allocation modulo the symmetric assignment.

state	q_1	q_3	q_4	q_6	q_7	q_9	q_{10}	q_{12}	q_{13}	q_{33}
instructions	12	1	3	1	4	1	18	1	$\underline{5}$	1
state	q_{14}	q_{16}	q_{18}	q_{20}	q_{22}	q_{23}	q_{25}	q_{27}	q_{32}	q_h
instructions	6	3	3	3	1	$\underline{15}$	8	20	6	0

This gives a total of 112 instructions for a weakly universal generalized 2-register machine. To obtain weak-strong universality, the result has to be decoded into a third increment-only register, which means repeated division of the encoding by 7 with incrementing the new register in each cycle, iterated until a remainder is obtained. In fact, this means adding the following generalized register machine instructions:

$$
\begin{aligned}
&q_h : (\mathtt{SUB}(2), h_1, h_7),\\
&h_i : (\mathtt{SUB}(2), h_{i+1}, h_8), 1 \leq i \leq 5,\\
&h_6 : (\mathtt{SUB}(2), \mathtt{ADD}(1)\mathtt{ADD}(3)q_h, h_8),\\
&h_7 : (\mathtt{SUB}(1), \mathtt{ADD}(2)h_7, q_h)
\end{aligned}
$$

with h_8 being the new halting state. The computation ends up with empty register 2, but still some "garbage" in register 1, which can be erased by taking an additional rule $h_8 : (\mathtt{SUB}(1), h_8, h_9)$ and h_9 as the new halting state instead. Hence, in total this additional part costs 8 instructions for the decoding, plus an extra instruction to erase the rest of the encoding, i.e., the instruction labeled by h_8, resulting in the overall value for the generalized register machine instructions of 121 (with "cleaning") and 120 (without "cleaning"), respectively.

Yet for weak universality, several states and rules can be saved by simulating the weakly universal machine U_{20} of Korec, see [10], instead of the strongly universal register machine U_{22}. The weakly universal register machine U_{20} does not use register 3, so we no longer need to carry out division by 19. The difference is only in the simulation block, so only instructions associated with states q_{23} and q_{27} are affected, as well as q_{30} and q_{31} work on different registers, which does not affect the number of generalized instructions, and instructions q_{25} and q_{29} are no longer present. Like in case of the strongly universal register

machine, we embed the instructions q_{30} and q_{31} into the preceding zero-test of q_{20}.

We leave the same assignment of prime numbers to the registers and the same allocation of the main register, except that we reallocate q_{23} to register 1 and that we no longer have q_{25}. This leads to skipping recopying for all transitions except for $q_{13} \to q_1$ and $q_{16} \to q_{23}$; again, the associated numbers of instructions are underlined in the table below. The necessity of at least two recopyings can be argued by inspecting the cycles $q_1 - q_6 - q_4 - q_7 - q_9 - q_{10} - q_{13} - q_1$ and $q_{16} - q_{18} - q_{27} - q_{32} - q_{23} - q_{16}$; these cycles have no common nodes, and each cycle needs at least one recopying to have the value in the original register. This minimality has been further confirmed by computer search in the space of possible allocations of registers to states, furthermore again showing the uniqueness of the optimal allocation modulo the symmetric assignment for the weakly universal generalized register machine with embedded increments.

We have the following adjustment on the number of generalized instructions; the numbers to the left of each arrow are replaced by the numbers to the right of that arrow.

state	q_{16}	q_{23}	q_{25}	q_{27}
instructions	$3 \to \underline{4}$	$\underline{15} \to 8$	$8 \to 0$	$20 \to 14$

After having saved 20 instructions in this way, only $112 - 20 = 92$ generalized instructions remain.

3.4 Generalized Counter Automata

Generalized counter automata (GCAs for short) were introduced in [1] and also used in [4] and [3] with slightly different restrictions because of how they then were simulated by the corresponding P systems. The reason to consider a generalization of counter automata is that sometimes the simulation costs (measured in the number of rules in the description of a P system) of a composite instruction is the same as that (or almost the same, but anyway less than the sum of those) for simulating an elementary instruction.

For a register machine $M = (m, B, l_0, l_h, P)$ consider the more general type of instructions $i : (q, M_-, N, M_+, q')$ where $q, q' \in Q$ are states, $N \subseteq R$ is a set of registers, and M_-, M_+ are multisets of registers. Such a register machine applies instruction i as follows: first, multiset M_- is subtracted from the register values (i.e., for each register $j \in R$, $M_-(j)$ is subtracted from the contents of register j; if at least one resulting value would be negative, the machine is blocked without producing any result); second, the subset N of registers is checked to be zero (if at least one of them is found to be non-zero, the machine is blocked without producing any result); third, the multiset M_+ is added to the register values (i.e., for each register $j \in R$, $M_+(j)$ is added to the contents of register j), and finally the state changes to q'.

The work of such a register machine, now also called a *generalized counter automaton* and written $M = (m, B, q_1, q_h, P)$, consists of derivation steps applying instructions, chosen in a non-deterministic way, associated with the current state. The computation starts in the initial state q_1, and we say that it halts if the final state q_h has been reached (which replaces the condition of reaching the final HALT-instruction labeled by l_h).

We start by presenting the small universal antiport P systems with inhibitors from [8]; let us call it GCA 1.

$1 : (q_1, \langle 1\rangle, \{\}, \langle 7\rangle, q_1)$,
$2 : (q_1, \langle\rangle, \{1\}, \langle 6\rangle, q_4)$,
$3 : (q_4, \langle 5\rangle, \{\}, \langle 6\rangle, q_4)$,
$4 : (q_4, \langle 6\rangle, \{5\}, \langle 5\rangle, q_{10})$,
$5 : (q_{10}, \langle 7, 6\rangle, \{\}, \langle 1, 5\rangle, q_{10})$,
$6 : (q_{10}, \langle 7\rangle, \{6\}, \langle 1\rangle, q_4)$,
$7 : (q_{10}, \langle\rangle, \{6, 7\}, \langle\rangle, q_1)$,
$8 : (q_{10}, \langle 6, 4\rangle, \{7\}, \langle\rangle, q_1)$,
$9 : (q_{10}, \langle 6, 5\rangle, \{7, 4\}, \langle\rangle, q_{18})$,
$10 : (q_{18}), \langle 5^3\rangle, \{\}, \langle 4\rangle, q_{18})$,
$11 : (q_{18}, \langle\rangle, \{5, 3\}, \langle 0\rangle, q_1)$,
$12 : (q_{18}, \langle 5^2, 0\rangle, \{5, 2\}, \langle\rangle, q_1)$,
$13 : (q_{18}, \langle 5^2, 2\rangle, \{5\}, \langle\rangle, q_1)$,
$14 : (q_{18}, \langle 5^2\rangle, \{5, 2, 0\}, \langle\rangle, q_1)$
$15 : (q_{18}, \langle 3, 4\rangle, \{5\}, \langle\rangle, q_1)$,
$16 : (q_{18}, \langle 5, 4\rangle, \{5\}, \langle 2, 3\rangle, q_1)$.

We now present a few variations of GCA 1, which have more instructions, but satisfy certain requirements that make them more suitable for a simulation by specific P systems.

For the variant to be simulated by anti-matter P systems, we require that for any instruction, M_- does not overlap with M_+; note that

this condition is already fulfilled by GCA 1. Moreover, we note that, as it will be shown later, if M_- does not overlap with N, then the simulation (in terms of the number of instructions) is more efficient, but otherwise the simulation is still more efficient than in the case of splitting such an instruction into two instructions and simulating these two. Another requirement, due to the technicalities of the simulation, is that the halting must be in a state with no associated instructions (unlike in GCA 1, which halts in q_{18} if no instruction is applicable, its straightforward simulation would non-deterministically choose an instruction to simulate and fail, entering an infinite loop). The solution is to replace the last two rules with the following ones; let us call this resulting automaton GCA 2:

$$15 : (q_{18}, \langle 3\rangle, \{5\}, \langle\rangle, q_{32}), \qquad 16 : (q_{18}, \langle 5\rangle, \{5\}, \langle 2, 3\rangle, q_{32}),$$
$$17 : (q_{32}, \langle 4\rangle, \{\}, \langle\rangle, q_1), \qquad 18 : (q_{32}, \langle\rangle, \{4\}, \langle\rangle, q_h).$$

For the simulation with many catalysts, we need different requirements. However, also in this case we need that the GCA halts in a state with no associated instructions, so we take GCA 2 as the basis. While it no longer matters whether M_- and N are disjoint, we require that M_+ does not overlap with either M_- or N. To fulfill this condition, we take GCA 2 and replace instruction 4 by new instructions 4 and $4'$ below. Let us call the result GCA 3. Moreover, for technical reasons, we have to produce the output in a register that only has increment instructions associated to it, and have all other registers empty in the end, hence, instruction 18 is replaced by instructions 18–21 below. Let us call the result GCA 4.

$$4 : (q_4, \langle 6\rangle, \{5\}, \langle\rangle, q_{4'}) \qquad 4' : (q_{4'}, \langle\rangle, \{\}, \langle 5\rangle, q_{10}),$$
$$18 : (q_{32}, \langle 0\rangle, \{4\}, \langle 8\rangle, q_{32}), \qquad 20 : (q_{32}, \langle 6\rangle, \{4\}, \langle\rangle, q_{32}),$$
$$19 : (q_{32}, \langle 1\rangle, \{4\}, \langle\rangle, q_{32}), \qquad 21 : (q_{32}, \langle\rangle, \{0, 1, 4, 6\}, \langle\rangle, q_h).$$

Finally, for a simulation with multiple catalysts (in fact, 8), the setting is more restricted. A coupling function f_c is considered, which is a bijective mapping from the set of registers to the same set, without a fixed point. Not only is M_- forbidden to contain more than one copy of the same register, but we need all the sets $supp(M_-)$, $f_c(supp(M_-)$

and N to be disjoint. After having carefully inspected the Korec machines and the resulting GCAs from [4], we decided to use the following coupling function f_c:

r:	0	1	2	3	4	5	6	7
$f_c(r)$	6	5	7	4	3	1	0	2

While we keep instructions 1–9 identical to the ones listed above, the rest of instructions is presented below. We call the result GCA 5 (in [3] such a variant of counter automata is called "weakly generalized").

$10 : (q_{18}, \langle 5\rangle, \{\}, \langle\rangle, q_{20})$,
$10' : (q_{20}, \langle 5\rangle, \{\}, \langle 4\rangle, q_{16})$,
$10'' : (q_{16}, \langle 5\rangle, \{\}, \langle\rangle, q_{18})$,
$11 : (q_{18}, \langle\rangle, \{3, 5\}, \langle 0\rangle, q_1)$,
$12 : (q_{16}, \langle 0\rangle, \{2, 5\}, \langle\rangle, q_1)$,
$13 : (q_{16}, \langle 2\rangle, \{5\}, \langle\rangle, q_{32})$,
$14 : (q_{16}, \langle\rangle, \{0, 2, 5\}, \langle\rangle, q_{32})$,
$15 : (q_{18}, \langle 3\rangle, \{5\}, \langle\rangle, q_{32})$,
$16 : (q_{20}, \langle\rangle, \{5\}, \langle 2, 3\rangle, q_{32})$,
$17 : (q_{32}, \langle 4\rangle, \{\}, \langle\rangle, q_1)$,
$18 : (q_{32}, \langle\rangle, \{4\}, \langle\rangle, q_h)$.

As in the case of multiple catalysts, the input must be moved to an increment-only register, but for technical reasons the other registers do not have to be cleaned by instructions of the GCA. Hence, we replace instruction 18 by the instructions below, with λ being the new final state, and we call the result GCA 6.

$$18 : (q_{32}, \langle 0\rangle, \{4\}, \langle 8\rangle, q_{32}), \qquad 18' : (q_{32}, \langle\rangle, \{0, 4\}, \langle\rangle, \lambda).$$

4 Antimatter

We now consider P systems with matter/anti-matter annihilation rules, see [1].

Theorem 1 *(see [1]) There exist small universal P systems with non-cooperative rules and matter/anti-matter annihilation rules – with 9 annihilation rules and, in total,* **53** *rules in the accepting case,* 59 *rules in the generating case, and* 57 *rules in the computing case.*

$$\begin{aligned}\Pi &= \big(O,\ [\ ]_1\ , q_1, R_1, 1, 1\big) \text{ where}\\ O &= \{l_2, l_4, l_6, l_7, l_8, l_9, l_{11}, l_{12}, l'_{12}, l_{13}, l'_{13}, l_{14}, l'_{14}, l_{15}, l_{16}, l'_{16}, l_{18}\}\\ &\cup\ \{q_1, q_4, q_{10}, q_{18}, q_{32}, q_h\} \cup \{a, a^- \mid a \in \{a_j \mid 0 \le j \le 7\} \cup \{\#\}\}\end{aligned}$$

and R_1 contains the following rules:

$$\begin{array}{lll}
q_1 \to q_1 {a_1}^- a_7, & & \\
q_1 \to l_2 {a_1}^-, & l_2 \to q_4 \# a_6, & \\
q_4 \to q_4 {a_5}^- a_6, & & \\
q_4 \to l_4 {a_5}^-, & l_4 \to q_{10} \# {a_6}^- a_5, & \\
q_{10} \to q_{10} {a_7}^- {a_6}^- a_1 a_5, & & \\
q_{10} \to l_6 {a_6}^-, & l_6 \to q_4 \# {a_7}^- a_1, & \\
q_{10} \to l_7 {a_6}^- {a_7}^-, & l_7 \to q_1 \#\#, & \\
q_{10} \to l_8 {a_7}^-, & l_8 \to q_1 \# {a_6}^- {a_4}^-, & \\
q_{10} \to l_9 {a_7}^- {a_4}^-, & l_9 \to q_{18} \#\# {a_6}^- {a_5}^-, & \\
q_{18} \to q_{18} {a_5}^- {a_5}^- {a_5}^- a_4, & & \\
q_{18} \to l_{11} {a_5}^- {a_3}^-, & l_{11} \to q_1 \#\# a_0, & \\
q_{18} \to l_{12} {a_5}^- {a_5}^- a_0^-, & l_{12} \to l'_7 {a_5}^- {a_2}^-, & \\
q_{18} \to l_{13} {a_5}^- {a_5}^- {a_2}^-, & l_{13} \to l'_{13} {a_5}^-, & l'_{13} \to q_1 \#, \\
q_{18} \to l_{14} {a_5}^- {a_5}^-, & l_{14} \to l'_{14} {a_5}^- {a_2}^- {a_0}^-, & l'_{14} \to q_1 \#\#\#, \\
q_{18} \to l_{15} {a_5}^-, & l_{15} \to q_{32} \# {a_3}^-, & \\
q_{18} \to l_{16} {a_5}^-, & l_{16} \to l'_{16} {a_5}^-, & l'_{16} \to q_{32} \# a_2 a_3, \\
q_{32} \to q_1 {a_4}^-, & & \\
q_{32} \to l_{18} {a_4}^-, & l_{18} \to q_h \#, & \\
\#^- \to \#^4, & \# \to \#^4, & (\#\#^- \to \lambda), \\
{a_r}^- \to \#^-, & (a_r {a_r}^- \to \lambda), & 0 \le r \le 7.
\end{array}$$

Table 1. A small universal P system with anti-matter.

For a generalized counter automaton $M = (m, B, q_1, q_h, P)$, let

$$k = 1 + \max_{i:(q, M_-, N, M_+, q') \in P} (|M_-|, |N|).$$

Common for different instructions of M, we consider the following rules:

$$\#^- \to \#^k,\ \# \to \#^k,\ \#\#^- \to \lambda,\ a_r \to \#^-,\ a_r a_r^- \to \lambda,\ r \in R.$$

We recall the main construction block: the simulation of instruction $i : (q, M_-, N, M_+, q') \in P$. First we consider the case when M_- and N have no common elements, and moreover, we also assume that M_- does not overlap with M_+.

$$q \to l_i \prod_{r \in N} {a_r}^-,\ l_i \to q'(\prod_{r \in N} \#)(\prod_{r \in M_-} {a_r}^-) \prod_{r \in M_+} a_r.$$

If the zero-test set N is empty, then the first step is a simple renaming and can be combined with the second step, yielding one rule

$$q \to q'(\prod_{r \in M_-} {a_r}^-) \prod_{r \in M_+} a_r.$$

Clearly, if M_- and N overlap, such an instruction can be broken down into two subsequent instructions of the generalized counter automaton. However, a more efficient solution with only three rules exists:

$$q \to l_i \prod_{r \in M_-} {a_r}^-,\ l_i \to l_i' \prod_{r \in N} {a_r}^-,\ l_i' \to q(\prod_{r \in N} \#) \prod_{r \in M_+} a_r.$$

The accepting case is shown by the construction in Table 1, simulating GCA 2.

5 One Catalyst and One Multi-Stable Catalyst

A conditional decrement is performed by letting the multi-stable catalyst try to remove one register object, the states of the catalyst being associated to the registers. In the next step, the "program object" verifies whether the state of the multi-stable object was changed, and the proper transition is modeled. Based on this idea, a few universal P systems have been constructed in [4], depending on whether the strong Korec machine, the weak one, or the one reduced to two working registers is simulated, whether the output is decoded, and whether the feature of toxic objects is used, see the upper part of Table 4.

6 Multiple Catalysts

In this section, we do not limit the number of catalysts, but aim at a small number of rules, based on the results recently established in [3].

$$
\begin{aligned}
\Pi &= (O, \Sigma, C = \{c_r \mid 0 \le r \le 7\}, \mu = [\ ]_1\ , w_1, R_1, f = 1),\\
O &= \{o_r, d_r, e_r \mid 0 \le r \le 7\} \cup \{\#, p_{10'}, p_{10''}, p_{18'}, o_8\}\\
&\cup \{p'_j \mid j \in \{1,3,4,5,6,8,9,10,10',10'',12,13,15,17,18'\}\}\\
&\cup \{p_j \mid 1 \le j \le 18\} \cup \{q_1, q_4, q_{10}, q_{16}, q_{18}, q_{20}, q_{32}\},\\
R_1 &= R \cup \{\# \to \#\} \cup \{c_r o_r \to c_r d_r,\ c_r d_r \to c_r,\ c_r e_r \to c_r \#,\\
&\quad c_{f_c(r)} e_r \to c_{f_c(r)},\ d_r \to \# \mid 0 \le r \le 7\},\\
w_1 &= q_1 d(),
\end{aligned}
$$

and the rules from the set R are listed below:

$q_1 \to p_1 e_1 d(1)$,	$p_1 \to p'_1 d(1,5)$,	$p'_1 \to q_1 d() o_7$,
$q_1 \to p_2 d(1)$,	$p_2 \to q_4 d() o_6$,	
$q_4 \to p_3 e_5 d(5)$,	$p_3 \to p'_3 d(1,5)$,	$p'_3 \to q_4 d() o_6$,
$q_4 \to p_4 e_6 d(5,6)$,	$p_4 \to p'_4 d(0,6)$,	$p'_4 \to q_{10} d() o_5$,
$q_{10} \to p_5 e_6 e_7 d(6,7)$,	$p_5 \to p'_5 d(0,2,6,7)$,	$p'_5 \to q_{10} d() o_1 o_5$,
$q_{10} \to p_6 e_7 d(6,7)$,	$p_6 \to p'_6 d(2,7)$,	$p'_6 \to q_4 d() o_1$,
$q_{10} \to p_7 d_{6,7}$,	$p_7 \to q_1 d()$,	
$q_{10} \to p_8 e_4 e_6 d(4,6,7)$,	$p_8 \to p'_8 d(0,3,4,6)$,	$p'_8 \to q_1 d()$,
$q_{10} \to p_9 e_5 e_6 d(4,5,6,7)$,	$p_9 \to p'_9 d(0,1,5,6)$,	$p'_9 \to q_{18} d()$,
$q_{18} \to p_{10} e_5 d(5)$,	$p_{10} \to p'_{10} d(1,5)$,	$p'_{10} \to q_{20} d()$,
$q_{20} \to p_{10'} e_5 d(5)$,	$p_{10'} \to p'_{10'} d(1,5)$,	$p'_{10'} \to q_{16} d()$,
$q_{16} \to p_{10''} e_5 d(5)$,	$p_{10''} \to p'_{10''} d(1,5)$,	$p'_{10''} \to q_{18} d()$,
$q_{18} \to p_{11} d(3,5)$,	$p_{11} \to q_1 d() o_0$,	
$q_{16} \to p_{12} e_0 d(0,2,5)$,	$p_{12} \to p'_{12} d(0,6)$,	$p'_{12} \to q_1 d()$,
$q_{16} \to p_{13} e_2 d(2,5)$,	$p_{13} \to p'_{13} d(2,7)$,	$p'_{13} \to q_{32} d()$,
$q_{16} \to p_{14} d(0,2,5)$,	$p_{14} \to q_{32} d()$,	
$q_{18} \to p_{15} e_3 d(3,5)$,	$p_{15} \to p'_{15} d(3,4)$,	$p'_{15} \to q_{32} d()$,
$q_{20} \to p_{16} d(5)$,	$p_{16} \to q_{32} d() o_2 o_3$,	
$q_{32} \to p_{17} e_4 d(4)$,	$p_{17} \to p'_{17} d(3,4)$,	$p'_{17} \to q_1 d()$,
$q_{32} \to p_{18} e_0 d(0,4)$,	$p_{18} \to p'_{18} d(0,6)$,	$p'_{18} \to q_{32} d() o_8$,
$q_{32} \to p_{18'} d(0,4)$,	$p_{18'} \to d()$.	

Table 2. A universal catalytic P system with 8 catalysts.

Theorem 2 *(see [3]) There exists a small universal catalytic P system with 8 catalysts and* **98** *rules. Using toxic objects, the number of rules can be reduced to 89.*

Besides the rules associated to the instructions, we use rules

$$\{\# \to \#\} \cup \{c_r o_r \to c_r d_r,\ c_r d_r \to c_r,\ c_r e_r \to c_r \#,$$
$$c_{f_c(r)} e_r \to c_{f_c(r)},\ d_r \to \# \mid 0 \le r \le 7\}.$$

For a general instruction j of wGCA, $j : (q_i, M_-, N, M_+, q_k)$ it suffices to have the following three rules:

$q_i \to p_j E_{M_-} D_{m,M_-,N}$, $p_j \to p_j D'_{m,M_-}$, $p_j \to q_k D_m O_{M_+}$ where

$$\begin{aligned} D_{m,M_-,N} &= \prod_{i \in [1..m] \setminus (supp(M_-) \cup N)} d_i, \\ D'_{m,M_-} &= \prod_{i \in [1..m] \setminus \{r, c(r) \mid r \in M_-\}} d_i, \\ E_{M_-} &= \prod_{r \in M_-} e_r, \text{ and} \\ O_{M_+} &= \prod_{r \in M_+} o_r. \end{aligned}$$

The construction given in Table 2 was used for the proof, simulating GCA 6 (for conciseness, the multiset of objects d_r, $0 \le r \le 7$, $r \notin M$, is denoted by $d(M)$, and we omit the braces denoting M).

In addition to w_1, to the initial configuration we add the number of symbols o_1 corresponding with the code of the machine to be simulated and the number of symbols o_0 corresponding with the input number to this machine; the result of the simulation is represented by the number of symbols o_8 in the final configuration.

Going to the extreme with the number of catalysts, we can even obtain a real-time simulation of register machines, see [4], and obtain a universal P system with even less rules.

$$
\begin{aligned}
\Pi &= (O, C, \{o_1, o_2\}, \{o_8\}, w, R) \text{ where} \\
O &= C \cup \{o_r \mid 0 \le r \le 8\} \cup Q \cup \{d_{r,-},\ d_{r,0} \mid 0 \le i \le 7\} \cup \{d, d', \#\}, \\
C &= \{c_{r,-},\ c_{r,0} \mid 0 \le r \le 7\} \cup \{c_d, c_p, c_\#\}, \\
w &= c_{0,-} \cdots\ c_{4,-}\ c_{5,-}\ c_{5,-}\ c_{5,-}\ c_{6,-}\ c_{7,-}\ c_{0,0}\ \cdots\ c_{7,0}\ \ c_d c_p c_\# \\
&\quad dd' p_1 e_S(d_{1,-}), \text{and the set } R \text{ consists of the following rules:} \\
R &= \{c_{r,-} o_r \to c_{r,-},\ c_{r,-} d \to c_{r,-}\# \mid r \in \{0, \cdots, 7\}\} \\
&\cup\ \{c_{r,0} o_r \to c_{r,0}\#,\ c_{r,0} d_{r,0} \to c_{r,0},\ c_{r,-} d_{r,-} \to c_{r,-}, \\
&\quad c_\# d_{r,-} \to c_\#\# \mid r \in \{0, \cdots, 7\}\} \\
&\cup\ \{c_d d' \to c_d,\ c_d d \to c_d,\ c_\# d' \to c_\#\#,\ c_\#\# \to c_\#\#\} \\
&\cup\ \{c_p q_1 \to c_p q_1 d' e_S(d_{1,-}) o_7,\ c_p q_1 \to c_p q_4 d' e_S(d_{1,0}) o_6, \\
&\quad c_p q_4 \to c_p q_4 d' e_S(d_{5,-}) o_6,\ c_p q_4 \to c_p q_{4'} d' e_S(d_{6,-}\, d_{5,0}), \\
&\quad c_p q_{4'} \to c_p q_{10} d' e_S(\lambda) o_5, \\
&\quad c_p q_{10} \to c_p q_{10} d' e_S(d_{6,-}\, d_{7,-}) o_1 o_5, \\
&\quad c_p q_{10} \to c_p q_4 d' e_S(d_{7,-}\, d_{6,0}) o_1, \\
&\quad c_p q_{10} \to c_p q_1 d' e_S(d_{6,0}\, d_{7,0}),\ c_p q_{10} \to c_p q_1 d' e_S(d_{4,-}\, d_{6,-}\, d_{7,0}), \\
&\quad c_p q_{10} \to c_p q_{18} d' e_S(d_{5,-}\, d_{6,-}\, d_{4,0}\, d_{7,0}), \\
&\quad c_p q_{18} \to c_p q_{18} d' e_S(d_{5,-}\, d_{5,-}\, d_{5,-}) o_4, \\
&\quad c_p q_{18} \to c_p q_1 d' e_S(d_{3,0}\, d_{5,0}) o_0, \\
&\quad c_p q_{18} \to c_p q_1 d' e_S(d_{0,-}\, d_{5,-}\, d_{5,-}\, d_{2,0}\, d_{5,0}) o_4, \\
&\quad c_p q_{18} \to c_p q_{32} d' e_S(d_{2,-}\, d_{5,-}\, d_{5,-}\, d_{5,0}) o_4, \\
&\quad c_p q_{18} \to c_p q_{32} d' e_S(d_{5,-}\, d_{5,-}\, d_{0,0}\, d_{2,0}\, d_{5,0}) o_4, \\
&\quad c_p q_{18} \to c_p q_{32} d' e_S(d_{3,-}\, d_{5,0}), \\
&\quad c_p q_{18} \to c_p q_{32} d' e_S(d_{5,-}\, d_{5,0}) o_2 o_3, c_p q_{32} \to c_p q_1 d' e_S(d_{4,-}), \\
&\quad c_p q_{32} \to c_p q_{32} d' e_S(d_{0,-}\, d_{4,0}) o_8, \\
&\quad c_p q_{32} \to c_p q_{32} d' e_S(d_{1,-}\, d_{4,0}),\ c_p q_{32} \to c_p q_{32} d' e_S(d_{6,-}\, d_{4,0}), \\
&\quad c_p q_{32} \to c_p e_S(d_{0,0}\, d_{1,0}\, d_{4,0}\, d_{6,0})\}.
\end{aligned}
$$

Table 3. A universal purely catalytic P system with 21 catalysts.

Theorem 3 *(see [4]) There exists a small universal purely catalytic P system with 21 catalysts and* **74** *rules. Using toxic objects, the number of rules can be reduced to 64.*

Let S be a finite multiset and $S' \subseteq S$, and let $e_S(S')$ be a string representing the multiset $S \setminus S'$. We note that we will use $e_S(\lambda)$ (as λ denotes the empty multiset) for representing the multiset S itself. We define the multiset S and the corresponding mapping e_S by

$$e_S(\lambda) = d_{0,-} \cdots d_{4,-}\ d_{5,-}\ d_{5,-}\ d_{5,-}\ d_{6,-}\ d_{7,-}\ d_{0,0}\ \cdots\ d_{7,0},$$

and for a finite multiset L, by $g(L)$ we denote a string representing a multiset consisting of objects o_r for each occurrence of r in L. Besides the rules associated to instructions, the following rules are used:

$$\begin{aligned} R &= \{c_{r,-}o_r \to c_{r,-},\ c_{r,-}d \to c_{r,-}\# \mid r \in \{0,\cdots,7\}\} \\ &\cup \{c_{r,0}o_r \to c_{r,0}\#,\ c_{r,0}d_{r,0} \to c_{r,0},\ c_{r,-}d_{r,-} \to c_{r,-}, \\ &\quad c_\# d_{r,-} \to c_\# \# \mid r \in \{0,\cdots,7\}\} \\ &\cup \{c_d d' \to c_d,\ c_d d \to c_d,\ c_\# d' \to c_\# \#,\ c_\# \# \to c_\# \#\}. \end{aligned}$$

If we take S to be the finite multiset over $\{d_{r,-}\,,\ d_{r,0} \mid 1 \leq r \leq m\}$ such that, for $1 \leq r \leq m$, $S(d_{r,0}) = 1$ if $r \in \bigcup_{j:(q,M_-,N,M_+,q')\in P} N$ and $S(d_{r,0}) = 0$ otherwise, as well as $S(d_{r,-}) = \max\{M_-(r) \mid j : (q, M_-, N, M_+, q') \in P\}$, then the simulation of an instruction $j : (q, M_-, N, M_+, q')$ is initiated by the catalytic rule

$$c_p q \to c_p q' d' e_S(\langle {d_{r,-}}^{M_-(r)} \mid r \in supp(M_-)\rangle \cup \langle d_{r,0} \mid r \in N\rangle) g(M_+).$$

The construction for the proof simulating GCA 4 is given in Table 3.

7 Universal P Systems with Two Catalysts

We now take the new simulation from [3]. For a register machine with only two working registers, we need 5 rules per instruction plus 11 rules; cleaning happens by the P system itself at the end of a successful

simulation (for example, see [16], for detailed arguments), but recopying of the result to an extra non-decrementable register at the end of the simulation is needed for the case of weak universality. Hence, we obtain a weakly-strongly / weakly universal catalytic P system with two catalysts, having **611**/**476** rules (improving the result of 1091/848 rules from [4]). Using the feature of toxic objects, the simulation costs are reduced to 5 rules per instruction plus 8, yielding **608**/**473** rules (improving the result of 726/564 rules from [4]).

8 Universal Purely Catalytic P Systems

It was stated in [3] that the constructions obtained there for cat_m also hold for $pcat_{m+2}$: one catalyst can take care of the states and program symbols, while one more catalyst can perform the trapping rules. Hence, any generalized register machine with m decrementable registers and s generalized SUB-instructions can be simulated by a *purely* catalytic P system with $m+2$ catalysts and $5s+5m+1$ rules. Therefore, the results with 611, 476, 608 and 473 rules for universal catalytic P systems with 2 catalysts also hold for universal purely catalytic P systems with 4 catalysts.

It was shown in [3] that purely catalytic P systems with 9 catalysts are strongly universal with $6 \times 16 + 6 \times 8 + 1 = 145$ rules. Using the formula $6s + 6m + 1$ from [17], simulating the weakly universal generalized register machine we obtain a weakly universal purely catalytic P system with 8 catalysts and $6 \times 15 + 6 \times 7 + 1 = \mathbf{133}$ rules (improving the result of 171 rules from [4]). In a similar approach, consider the weakly-strongly/weakly universal generalized register machine with $m = 2$ decrementable registers and $s = 93/s = 120$ generalized register machine instructions. Again using the formula $6s + 6m + 1$ from the recent paper [17], we obtain a weakly-strongly/weakly universal purely catalytic P system with 3 catalysts and $6 \times 120 + 6 \times 2 + 1 = \mathbf{733}/6 \times 93 + 6 \times 2 + 1 = \mathbf{571}$ rules, thus improving the previously best known results of 1091/848 rules, respectively.

9 Conclusions

It has been known that only one bi-stable catalyst suffices for computational completeness of P systems (having non-cooperative rules besides the bi-catalytic ones) and that purely catalytic P systems with three catalysts are computationally complete. With two catalysts computational completeness can be obtained if one of them is bi-stable, see [4].

Feature	s	ws	w	s,tox	ws,tox	w,tox
$p8cat$, $pcat$	$61_{[4]}$			$47_{[4]}$		
$p7cat$, $pcat$			$56_{[4]}$			$43_{[4]}$
$p2cat$, $pcat$		$483_{[4]}$	$371_{[4]}$		$362_{[4]}$	$278_{[4]}$
$pcat_{21}$	$74_{[4]}$					
$pcat_{20}$				$64_{[4]}$		
$pcat_{10}$	$98_{[3]}$			$89_{[3]}$		
cat_8	$98_{[3]}$			$89_{[3]}$		
$pcat_9$	$145_{[3]}$					
$\mathbf{pcat_8}$			$\mathbf{133}_{[17]+[4]}$	$120_{[4]}$		
$pcat_7$						$111_{[4]}$
$\mathbf{pcat_4}$		$\mathbf{611}_{+[4]}$	$\mathbf{476}_{+[4]}$		$\mathbf{608}_{+[4]}$	$\mathbf{473}_{+[4]}$
$\mathbf{cat_2}$		$\mathbf{611}_{+[4]}$	$\mathbf{476}_{+[4]}$		$\mathbf{608}_{+[4]}$	$\mathbf{473}_{+[4]}$
$\mathbf{pcat_3}$		$\mathbf{733}_{[17]+[4]}$	$\mathbf{571}_{[17]+[4]}$			
$pcat_2$					$726_{[4]}$	$564_{[4]}$

Table 4. Number of rules in universal catalytic P systems. We write "s" for strongly universal P systems, "ws" for weakly-strongly universal P systems and "w" for weakly universal P systems, "tox" for P systems with toxic objects, "cat_k" for k catalysts, $mcat$ for an m-stable catalyst, and "p" indicates P systems without non-cooperative rules.

Generalizing counter automata by allowing them to perform multiple operations on multiple registers, a few small generalized counter

automata are obtained (from 16 rules to 22 rules), depending on the specific requirements of P systems that would simulate them. Generalized counter automata are a very convenient tool for constructing small universal P systems. For instance, small strongly universal P systems with anti-matter with 9 annihilation rules and, in total, **53** rules in the accepting case, 59 rules in the generating case, and 57 rules in the computing case can be constructed.

By optimizing the reduction of the universal register machines U_{22} and U_{20} to register machines with two working registers, in [4] a strongly universal register machine with 120 instructions and two decrementable registers and a weakly universal register machine with 92 instructions and two registers have been obtained.

The now best known results for catalytic systems are summarized in Table 4. It describes universal (purely or not) catalytic P systems with and without toxic objects where the type of universality ranges from strong over weak-strong to weak. The results in the upper part of the table correspond to one normal catalyst and one m-stable catalyst, $2 \leq m \leq 8$, while the results in the lower part of the table correspond to k catalysts, $2 \leq k \leq 21$. Depending on all these features, the overall number of rules varies from 43, top right, to 733, bottom left.

The new results elaborated in this paper are indicated in boldface. If some entry of a table contains "+", then the reference following it indicates where the underlying simulating model has been studied, while the reference preceding "+" (if indicated, otherwise we imply the current paper) indicates where the currently best known simulated complexity has been obtained. Three small universal P systems, namely, the one with anti-matter and 53 rules, the catalytic one with 8 catalysts and 98 rules, and the purely catalytic one with 21 catalysts and 74 rules, were chosen to be presented explicitly in Tables 1, 2, and 3.

References

[1] A. Alhazov, B. Aman, R. Freund, Gh. Păun: Matter and Anti-Matter in Membrane Systems. In: H. Jürgensen, J. Karhumäki, A. Okhotin (Eds.): *16th International Workshop on Descriptional Complexity of Formal Sys-*

tems, DCFS 2014, Lecture Notes in Computer Science **8614**, 2014, 65–76.

[2] A. Alhazov, R. Freund: P Systems with Toxic Objects. In: M. Gheorghe, G. Rozenberg, A. Salomaa, P. Sosík, C. Zandron: *Membrane Computing - 15th International Conference, CMC 2014, Prague*, Lecture Notes in Computer Science **8961**, 2014, 99–125.

[3] A. Alhazov, R. Freund: Small Catalytic P Systems. Workshop on Membrane Computing, Auckland, accepted, 2015.

[4] A. Alhazov, R. Freund: Variants of Small Universal P Systems with Catalysts. *Fundamenta Informaticae* **138**(1-2), 227–250, 2015.

[5] A. Alhazov, Yu. Rogozhin, S. Verlan: On Small Universal Splicing Systems. *International Journal of Foundations of Computer Science* **23** (7), 2012, 1423–1438.

[6] A. Alhazov, S. Verlan: Minimization Strategies for Maximally Parallel Multiset Rewriting Systems. *Theoretical Computer Science* **412** (17), 2011, 1581–1591.

[7] R. Freund, L. Kari, M. Oswald, P. Sosík: Computationally Universal P Systems without Priorities: Two Catalysts Are Sufficient. *Theoretical Computer Science* **330** (2), 2005, 251–266.

[8] R. Freund, M. Oswald: A Small Universal Antiport P System with Forbidden Context. In: H. Leung, G. Pighizzini (Eds.): *Proceedings of the 8th International Workshop on Descriptional Complexity of Formal Systems*, DCFS 2006, Las Cruces, New Mexico, USA, 2006, New Mexico State University, 2006, 259–266.

[9] S. Ivanov, E. Pelz, S. Verlan: Small Universal Non-deterministic Petri Nets with Inhibitor Arcs. In: H. Jürgensen, J. Karhumäki, A. Okhotin (Eds.): *16th International Workshop on Descriptional Complexity of Formal Systems, DCFS 2014*, Lecture Notes in Computer Science **8614**, 2014, 186–197.

[10] I. Korec: Small Universal Register Machines. *Theoretical Computer Science* **168**, 1996, 267–301.

[11] M.L. Minsky: *Computation: Finite and Infinite Machines*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1967.

[12] Gh. Păun: Computing with Membranes. *Journal of Computer and System Sciences* **61** (1), 108–143 (2000) (and Turku Center for Computer Science – TUCS Report 208, November 1998, `www.tucs.fi`).

[13] Gh. Păun: *Membrane Computing. An Introduction*. Springer, 2002.

[14] Gh. Păun, G. Rozenberg, A. Salomaa: *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.

[15] G. Rozenberg, A. Salomaa (Eds.): *Handbook of Formal Languages,* 3 volumes. Springer, 1997.

[16] P. Sosík, M. Langer: Improved Universality Proof for Catalytic P Systems and a Relation to Non-Semilinear Sets. In: S. Bensch, R. Freund, F. Otto (Eds.): *Sixth Workshop on Non-Classical Models of Automata and Applications (NCMA 2014)*, books@ocg.at, BAND 304, 2014, 223–233.

[17] P. Sosík, M. Langer: Small Catalytic P Systems Simulating Register Machines. *Theoretical Computer Science*, accepted, 2015.

[18] P systems webpage. `http://ppage.psystems.eu`.

Artiom Alhazov, Rudolf Freund, Petr Sosík Received July 13, 2015

Artiom Alhazov
Institute of Mathematics and Computer Science
Str. Academiei 5,
Chişinău, MD-2028, Moldova
E–mail: `artiom.alhazov@math.md`

Rudolf Freund
Faculty of Informatics, TU Wien
Favoritenstraße 9-11, 1040 Wien, Austria
E–mail: `rudi@emcc.at`

Petr Sosík
Research Institute of the IT4Innovations Centre of Excellence
Faculty of Philosophy and Science, Silesian University in Opava
74601 Opava, Czech Republic
E–mail: `petr.sosik@fpf.slu.cz`

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

BioMaxP : A Formal Approach for Cellular Ion Pumps

Bogdan Aman Gabriel Ciobanu

Abstract

We look at the living cells as complex systems of ion pumps working in parallel to ensure proper physiologic functionalities. To model such a system of pumps, we define a formalism called BioMaxP that allows working with multisets of ions, explicit interpretation of the transportation (from inside to outside, and from outside to inside) based on the number of existing ions, and a maximal parallel execution of the involved pumps.

1 Introduction

All living cells can be seen as complex systems of interacting components, having different concentrations of ions (e.g., Na^+, K^+ and Ca^{++}) across the cell membrane. Under resting conditions, Na^+ and Ca^{++} ions enter the cells and K^+ ions exit the cell because the concentration of K^+ is high inside the cell and low outside, while the opposite situation is found for Na^+ and Ca^{++}. A fundamental mechanism in most of the living cells is the Na^+/K^+-$ATPase$ that is essential for the maintenance of Na^+ and K^+ concentrations across the membrane by transporting Na^+ out of the cell and K^+ back into the cell. This pump is the first discovered ion transporter; for this discovery, the Danish chemist Jens Skou received the Nobel Prize in 1997.

In this paper we model the movement of ions and the conformational transformations of ion transporters (NaK ion pumps, Na and K ion channels) using a very simple but powerful new formalism called

BioMaxP . We use an operational semantics able to capture quantitative aspects (e.g., number of ions) and abstract conditions associated with evolution (e.g., the number of ions is between certain thresholds). The modelling aims to facilitate a better understanding of the living cell viewed as a complex system of parallel ion transporters.

The novelty of our approach is that we model systems composed of more than one pump as usually done (e.g., see [5]). Since the pumps non-deterministically choose which ions to transport, the complexity of such systems increases with the number of pumps. For further notions about NaK pump, the interested reader can consult [1].

2 Syntax and Semantics of BioMaxP

The prototyping language BioMaxP provides sufficient expressiveness to model in an elegant way the interaction in complex systems of parallel ion pumps. The cell is a complex system of parallel pumps trying to keep the equilibrium of ions inside the cell. In order to model these pumps, we enforce that their functioning takes place only if the number of various types of ions is between some accepted limits given by min and max values. Therefore the syntax and semantics emphasize the process of counting them, and the way the quantities of ions vary during evolution. The semantics of BioMaxP is provided by multiset labelled transitions in which multisets of actions are executed in parallel.

Syntax of BioMaxP The syntax of BioMaxP is given in Table 1, where the following are assumed:

- a set *Chan* of ion transportation channels a, and a set Id of process identifiers (each $id \in Id$ has its arity m_{id});
- for each $id \in Id$ there is a unique process $id(u_1, \ldots, u_{m_{id}} : T_1, \ldots, T_{m_{id}}) \stackrel{def}{=} P_{id}$, where the distinct variables u_i are parameters, and the T_i are ions types;
- v is a tuple of expressions built from values, variables and allowed operations;
- T represent ions types.

Table 1. BioMaxP Syntax

Processes			
P, Q	$::=$	$a^{min}!(v : T)$ then P ı	(sending)
		$a^{max}?(f(u : T))$ then P ı	(receiving)
		$id(v)$ ı	(recursion)
		$P \mid Q$	(parallel)

A constraint ^{min} associated with a sending action $a^{min}!(z : T)$ then P makes the channel a available for sending z units/ions of type T only if the total available quantity of ions of type t is greater than min. A constraint ^{max} associated to a receiving action $a^{max}?(x : T)$ then P along a channel a is activated only if the number of ions of the type T available is less than max. The function f of the receiving action can be either id (we often omit it), meaning that the received ions are to be transported, or add, meaning that the ions are received from some other process. The only variable binding constructor is $a^{max}?(u : T)$ then P; it binds the variable u within P. The free variables of a process P are denoted by $fv(P)$; for a process definition, is assumed that $fv(P_{id}) \subseteq \{u_1, \ldots, u_{m_{id}}\}$, where u_i are the process parameters. Processes are defined up-to an alpha-conversion, and $\{v/u\}P$ denotes P in which all free occurrences of the variable u are replaced by v, eventually after alpha-converting P in order to avoid clashes. Processes are further constructed from the parallel composition $P \mid Q$. A system of parallel pumps is represented as a process with some initial values for the numbers of ions.

Remark 1 *In order to focus on the local interaction aspects of* BioMaxP *, we abstract from arithmetical operations, considering by default that the simple ones (comparing, addition, subtraction) are included in the language.*

Operational Semantics of BioMaxP The operational semantics rules of BioMaxP is presented in Table 2. The multiset labelled transi-

tions of form $P \xrightarrow{\Lambda} P'$ use a multiset Λ to indicate the actions executed in parallel in one step. When the multiset Λ contains only one action λ, in order to simplify the notation, $P \xrightarrow{\{\lambda\}} P'$ is simply written as $P \xrightarrow{\lambda} P'$. We assume that in order to interact the processes can commute, namely $P \mid Q$ is the same process as $Q \mid P$.

Table 2. BioMaxP Operational Semantics

$$\text{(Com)} \quad \frac{v : T \quad and \quad min \leq |T| \leq max}{a^{min}!\langle v\rangle \text{ then } P \mid a^{max}?(f(u:T)) \text{ then } P' \xrightarrow{\{v/u\}} P \mid \{v/u\}P' \text{ and } |T| = |T| - v \text{ if } f = id \text{ or } |T| = |T| + v \text{ if } f = add}$$

$$\text{(Call)} \quad \frac{\{v/u\}P_{id} \xrightarrow{id} P'_{id}}{id(v) \xrightarrow{id} P'_{id}} \text{ where } id(v:T) \stackrel{def}{=} P_{id}$$

$$\text{(Par1)} \quad \frac{P_1 \xrightarrow{\Lambda_1} P'_1 \qquad P \not\rightarrow}{P_1 \mid P \xrightarrow{\Lambda_1} P'_1 \mid P} \qquad \text{(Par2)} \quad \frac{P_1 \xrightarrow{\Lambda_1} P'_1 \quad P_2 \xrightarrow{\Lambda_2} P'_2}{P_1 \mid P_2 \xrightarrow{\Lambda_1 \cup \Lambda_2} P'_1 \mid P'_2}$$

In rule (Com), an output process $a^{min}!\langle v\rangle$ then P succeeds in sending a tuple of values v over channel a to process $a^{max}?(u:T)$ then P if v has the same type T as u and if the number of ions of type T is between min and max, namely $v : T$ and $min \leq |T| \leq max$. Both processes continue to execute, the first one as P and the second one as $\{v/u\}P'$. Once the ions are send away, $f = id$, the number of ions of type T becomes $T - |v|$, while if they are received, $f = add$, then the number of ions of type T becomes $T + |v|$. Rule (Call) describes the evolution of a recursion process. Rules (Par1) and (Par2) are used to compose larger processes from smaller ones by putting them in parallel, and considering the union of multisets of actions. In rule (Par2), $P \not\rightarrow$ denotes a process P that cannot evolve. It can be noticed that in rule (Par2) we use negative premises: an activity is performed based on the absence of actions. This is due to the fact that sequencing

the evolution can only be defined using negative premises, as done for sequencing processes [6, 10].

Example 1 *The use of* BioMaxP *for specifying complex systems of pumps is illustrated by describing in an explicit way the molecular interactions and conformational transformations of a large system of ion transporters, namely Na^+K^+ ATPases and Na and K ion channels, that are concerned with the movement of sodium-potassium ions in and out of a cell whenever certain thresholds are verified. The system we consider is formed from n_1 NaK pumps, n_2 Na channels and n_3 K channels. Each pump i is modelled by three processes: one that models the interaction of the pump with the environment, one modelling the interaction with the cell and another one that models the transport of ions through the membrane. The molecular components are processes modelled as the ends of a channel (one end for input, and another for output), while the molecular interaction coincides with communication on channels.*

The initial system of pumps is described in BioMaxP *by:*

$$
\begin{array}{c}
Cell(NaEnv, KEnv, NaCell, KCell, AtP, ADP, P) = \\
\mid NaKPumpEnv(0) \mid NaKPumpCell(0) \mid NaKPump(0) \\
\cdots \mid NaKPumpEnv(n_1\text{-}1) \mid NaKPumpCell(n_1\text{-}1) \mid NaKPump(n_1\text{-}1) \\
\mid NaPumpEnv(n_1) \mid NaPumpCell(n_1) \mid NaPump(n_1) \\
\cdots \mid NaPumpEnv(n_1+n_2-1) \mid NaPumpCell(n_1+n_2-1) \mid NaPump(n_1+n_2-1) \\
\mid KPumpEnv(n_1+n_2) \mid KPumpCell(n_1+n_2) \mid KPump(n_1+n_2) \\
\cdots \mid KPumpEnv(n_1+n_2+n_3\text{-}1) \mid KPumpCell(n_1+n_2+n_3\text{-}1) \mid \\
KPump(n_1+n_2+n_3\text{-}1) \\
CreateATP \mid ConsumeADP
\end{array}
$$

We present in detail some of the above processes. The others are written in a similar manner.

- $Cell(NaCell, KCell, AtP, ADP, NaEnv, KEnv, P)$ *is the system in which several quantities of ions are initialized.*
- *Each NaK-ATPase is described by three processes:*
  - $NaKPumpEnv(id) = site2[id]^{160}?(add(y_{na} : NaEnv))$
    then $site2[id]^2!\langle 2K\rangle$
    then $p[id]^6?(add(y_p : P))$
    then $NaKPumpEnv(id)$

The environment site of the pump contains the channel $site2[id]$ *used for receiving three ions of* Na^+ *and also for sending two ions of* K^+*, and also the channel* $p[id]$ *for receiving the produced* P *molecules. The sending and receiving operations modify also the number of ions present in the system of pumps: e.g., when sending two* K^+ *ions, an operation of the form* $KEnv = Kenv - 2$ *is performed, while receiving the* $y_{na} : NaEnv$ *ions an operation of the form* $NaEnv = NaEnv + 3$ *is performed due to the add function that is used to add the amount of received ions to the corresponding multiset.*

* $NaKPumpCell(id) = site1[id]^{(12,1)}!\langle(3Na, ATP)\rangle$
  then $adp[id]^6?(add(x_{adp} : ADP))$
  then $site1[id]^{150}?(add(x_k : KCell))$
  then $NaKPumpCell(id)$

The cell site of the pump contains the channel $site1[id]$ *used for sending three ions of* Na^+ *and one* ATP *and also for receiving two ions of* K^+*, and also the channel* $adp[id]$ *for receiving the produced* ADP *molecules.*

* $NaKPump(id) = site1[id]^{(28,9)}?((x_{na} : NaCell, x_{atp} : ATP))$
  then $adp[id]^0!\langle ADP\rangle$
  then $site2[id]^{100}!\langle 2Na\rangle$
  then $site2[id]^6?(y_k : KEnv)$
  then $p[id]^0!\langle P\rangle$ then $site1[id]^{110}!\langle 2K\rangle$
  then $NaKPump(id)$

This process describes the evolution of the pump, namely the transport of Na *and* K *ions between the environment and the cell.*

3 Timed Automata

Due to their simplicity, timed automata, extended with integer variables, structured data types, user defined functions, and channel synchronization, have been used by several tools (e.g., UPPAAL) for the simulation and verification of timed automata [2]. In what follows we consider a particular case of timed automata, namely we ignore the

time aspects as they are not relevant to our approach and will refer to timed automata as automata.

Syntax Assume a finite set of integer variables $\mathcal{C}$ ranged over by x, y, ... standing for data, and a finite alphabet Σ ranged over by a, b, ... standing for actions. A constraint is a conjunctive formula of constraints of the form $x \sim m$ for $x \in \mathcal{C}$, $\sim \in \{\leq, <, ==, >, \geq\}$, and $m \in \mathbb{N}$. The set of constraints, ranged over by g, is denoted by $\mathcal{B}(\mathcal{C})$.

Definition 1 *An* **automaton** $\mathcal{A}$ *is a tuple* $\langle N, n_0, E \rangle$*, where*

- N *is a finite set of nodes;*
- n_0 *is the initial node;*
- $E \subseteq N \times \mathcal{B}(\mathcal{C}) \times \Sigma \times \mathbb{N}^{\mathcal{C}} \times N$ *is the set of edges.*

$n \xrightarrow{g,a,r} n'$ *is a shorthand notation for* $\langle n, g, a, r, n' \rangle \in E$*. r denotes fresh assignments to variables after the transition is performed.*

Networks of Automata A network of automata is the parallel composition $\mathcal{A}_1 \mid \ldots \mid \mathcal{A}_n$ of a set of automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ combined into a single system. Synchronous communication inside the network is by handshake synchronization of input and output actions. In this case, the action alphabet Σ consists of $a?$ symbols (for input actions), $a!$ symbols (for output actions), and τ symbols (for internal actions). A detailed example is found in [9]. A network can perform action transitions (following an enabled edge). An action transition is enabled if all guards on the corresponding edges are satisfied.

Let u, v, ... denote assignments mapping $\mathcal{C}$ to naturals $\mathbb{N}$. $g \models u$ means that the values u satisfy the guard g. Let n_i stand for the ith element of a node vector n, and $n[n'_i/n_i]$ for the vector n with n_i being substituted with n'_i. A network state is a pair $\langle n, u \rangle$, where n denotes a vector of current nodes of the network (one for each automaton), and u is an assignment storing the current values of all network integer variables.

Definition 2 *The operational semantics of an automaton is a transition system where states are pairs* $\langle n, u\rangle$ *and transitions are defined by the rules:*

- $\langle n, u\rangle \xrightarrow{\tau} \langle n[n'_i/n_i], u'\rangle$ *if* $n_i \xrightarrow{g,\tau,r} n'_i$, $g \models u$ *and* $u' = r[u]$;
- $\langle n, u\rangle \xrightarrow{\tau} \langle n[n'_i/n_i][n'_j/n_j], u'\rangle$ *if there exist* $i \neq j$ *such that*
  1. $n_i \xrightarrow{g_i,a?,r_i} n'_i$, $n_j \xrightarrow{g_j,a!,r_j} n'_j$, $g_i \wedge g_j \models u$,
  2. $u' = r_i[r_j[u]]$.

4 Relating BioMaxP to Automata

In order to use existing tools such as UPPAAL for the verification of complex systems of parallel pumps, we establish a relationship between BioMaxP and automata.

Building an automaton for each process: Given a process P without the parallel operator at the top level, we associate to it an automaton $\mathcal{A} = \langle N, n_0, E\rangle$, where $n_0 = l_0$, $N = \{l_0\}$, $E = \emptyset$. The initial values of the BioMaxP system composed of P are set as the initial values of the automaton $\mathcal{A}$. The nodes of the associated automata are labelled using a fresh label l, and an index such that the nodes are uniquely labelled in this automaton (we start with the index 0, and increment it when necessary). The components N and E are updated depending on the structure of process P:

- for $P = a^{min}!\langle v\rangle$ then P_1 we have
  - $N = N \cup \{l_{i+1}\}$ where $i = max\{j \mid l_j \in N\}$;
    - The added node l_{i+1} indicates the execution of the process P, leading to P_1.
  - $E = E \cup \{n, min \leq |T|, a!, , l_{i+1}\}$;
    - If $i > 0$ it means that the automaton already contains some edges, and the process P was launched from the then branch of a process P'. Since the translation is made depending on the structure of the processes, it means that the action leading to P is already modelled

in the automaton. If $P' = b^{min'}!\langle w\rangle$ then P or $P' = b^{max'}?(u:T')$ then P, then the action of P' is modelled by an edge with the last component l_k, and thus $n = l_k$.
  - Otherwise, $n = l_0$.

  The edge encodes the then branch leading to process P_1. Channel a is an urgent channel (communication takes place as soon as possible).
- for $P = a^{max}?(f(u:T))$ then P_1 we have
  - $N = N \cup \{l_{i+1}\}$ where $i = max\{j \mid l_j \in N\}$;
  - $E = \begin{cases} E \cup \{l_i, |T| \leq max, a!, |T| = |T| - |u|, l_{i+1}\}, \text{if } f = id; \\ E \cup \{l_i, |T| \leq max, a!, |T| = |T| + |u|, l_{i+1}\}, \text{if } f = add. \end{cases}$
  A similar reasoning as for the previous case. Depending on function f, ions are removed or added from the number representing the existing ions of type T.
- for $P = P_1 \mid \ldots \mid P_k$, $k > 1$, and P_j does not contain operator $\mid$ at top level, then
  - $N = N \cup \{l_{i+1}\}$ where $i = max\{j \mid l_j \in N\}$;
    - If P contains some indexed nodes l (namely $l_0, \ldots, l_i$), then add l_{i+1} to N.
  - $E = E \cup \{n, , a!, \{x = 0\}, l_{i+1}\}$;
    - If $i > 0$, using a similar argument as for the communication actions, it holds that $n = l_k$. We use a new channel labelled a as a broadcast channel, in order to start at the same time all the parallel processes from P.
    - Otherwise, $n = l_0$.

  The new edge leads to process P_1. For each of the other processes P_j, $j > 1$, a new automaton $\mathcal{A}_j = \langle N_j, n_{j0}, E_j, I_j\rangle$ is build, where:
    - $n_{j0} = l_0$; $N_j = \{l_0, l_1\}$; $E_j = \{l_0, , a?, \{x = 0\}, l_1\}$; $I_j(l_0) = \emptyset$.

  The automaton is constructed recursively using the definition of P_j.

Building an automaton for each process leads to the next result about the equivalence between a BioMaxP process P and its corresponding automaton $\mathcal{A}_P$ in state $\langle n_P, u_P\rangle$ (i.e., $(\mathcal{A}_P, \langle n_P, u_P\rangle)$). Their transition

systems differ not only in transitions, but also in states; thus, we adapt the notion of bisimilarity:

Definition 3 *A symmetric relation $\sim$ between* BioMaxP *processes and their corresponding automata is a bisimulation if whenever* $(N, (\mathcal{A}_N, \langle n_N, u_N\rangle)) \in \sim$ *if* $P \xrightarrow{\lambda} P'$, *then* $\langle n_P, u_P\rangle \xrightarrow{\tau} \langle n_{P'}, u_{P'}\rangle$ *and* $(P', (\mathcal{A}_{P'}, \langle n_{P'}, u_{P'}\rangle)) \in \sim$ *for some* P'.

After defining bisimulation, we can state the following result.

Theorem 1 *Given a* BioMaxP *process* P, *there exists an automata* $\mathcal{A}_P$ *with a bisimilar behaviour. Formally,* $P \sim \mathcal{A}_P$.

Proof. [Sketch] The construction of the automaton simulating a given BioMaxP process is presented above. A bisimilar behaviour is given by the fact that a communication rule is matched by a synchronization between the edges obtained by translations. □

Thus, the size of an automata $\mathcal{A}_P$ is polynomial with respect to the size of a BioMaxP process P, and the state spaces have the same number of states.

Reachability Analysis. Qualitative properties abstract away from any quantitative information like time aspects or energy costs of targeted biological systems. One of the most useful question to ask about an automaton is the reachability of a given set of final states. Such final states may be used to characterize safety properties of a system.

Definition 4 *For an automata with initial state* $\langle n_0, u_0\rangle$, $\langle n, u\rangle$ *is reachable if and only if* $\langle n_0, u_0\rangle \xrightarrow{\tau}{}^{*} \langle n, u\rangle$. *More generally, given a constraint* $\phi \in \mathcal{B}(\mathcal{C})$ *if* $\langle n, u\rangle$ *is reachable for some* u *satisfying* ϕ, *then a state* $\langle n, \phi\rangle$ *is reachable.*

The reachability problem is decidable [4]. The reachability problem can be also defined for BioMaxP networks.

Definition 5 *Starting from a* BioMaxP *process* P_0*, a process* P_1 *is reachable if and only if* $P_0 \overset{\lambda}{\rightarrow}{}^* P_1$.

The following result is a consequence of Theorem 1.

Corollary 1 *For a* BioMaxP *process, the reachability problem is decidable.*

5 Conclusion

Previously, we provided a formal description of the sodium-potassium ion transport across cell membranes in terms of the π-calculus [8]. In [7], the transfer mechanisms were described step by step, and a software tool called Mobility Workbench [11] was used to verify some properties of the described system formed of only one pump. Inspired by the functioning of this pump, we introduced and studied a ratio-based type system using thresholds in a bio-inspired formalism [3]. The aim was to avoid errors in the definition of the formal models used to mimic the evolution of some biologic processes.

In this paper we try to unify and extend our previous attempts to model the movement of ions using the sodium-potassium-pump by introducing a very simple, elegant but powerful new formalism called BioMaxP able to capture the quantitative aspects (e.g., number of ions) and abstract conditions associated with evolution (e.g., the number of ions is between certain thresholds). This approach facilitates a better understanding of the processes happening in a cell viewed as a complex system of ion pumps working in parallel. The novelty is that we are able to model systems consisting of more than just a NaK pump by adding different amounts of other types of ion pumps.

Acknowledgements. The work was supported by a grant of the Romanian National Authority for Scientific Research, project number PN-II-ID-PCE-2011-3-0919.

References

[1] B. Alberts, A. Johnson, J. Lewis, D. Morgan, M. Raff, K. Roberts, P. Walter. *Molecular Biology of the Cell*, 6th edition, Garland Science, New York (2014).

[2] R. Alur, D.L. Dill. A Theory of Timed Automata. *Theoretical Computer Science* **126**, 183–235 (1994).

[3] B. Aman, G. Ciobanu. Behavioural Types Inspired by Cellular Thresholds. *Lecture Notes in Computer Science* **8368**, 1–15 (2014).

[4] J. Bengtsson, W. Yi. Timed Automata: Semantics, Algorithms and Tools. *Lecture Notes in Computer Science* **3098**, 87–124 (2004).

[5] D. Besozzi, G. Ciobanu. A P System Description of the Sodium-Potassium Pump. *Lecture Notes in Computer Science* **3365**, 210–223 (2005).

[6] B. Bloom, S. Istrail, A.R. Meyer. Bisimulation Can't Be Traced: Preliminary Report. *In 15th ACM Symposium on Principles of Programming Languages*, 229–239, 1988.

[7] G. Ciobanu. Software Verification of the Biomolecular Systems. in *Modelling in Molecular Biology*, Natural Computing Series, Springer, 40–59 (2004).

[8] G. Ciobanu, V. Ciubotariu, B. Tanasă. A π-calculus Model of the Na Pump, *Genome Informatics* **13**, 469–472 (2002).

[9] T.A. Henzinger, X. Nicollin, J. Sifakis, S. Yovine. Symbolic Model Checking for Real-time Systems. *Information and Computation* **111**, 192–224 (1994).

[10] F. Moller. *Axioms for Concurrency*. PhD Thesis, Department of Computer Science, University of Edinburgh, 1989.

[11] B. Victor, F. Moller. The Mobility Workbench - A Tool for the π-Calculus. *Lecture Notes in Computer Science* **818**, 428–440 (1994).

Bogdan Aman, Gabriel Ciobanu Received July 21, 2015

Romanian Academy, Institute of Computer Science
Blvd. Carol I no.11, 700506 Iaşi, Romania
E–mail: baman@iit.tuiasi.ro, gabriel@info.uaic.ro

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Institution for Pure First-Order Composition-Nominative Logic

Alexey Chentsov, Mykola Nikitchenko

Abstract

This paper aims to present composition-nominative logics as institutions. To be informative, pure first-order composition-nominative logic is chosen. Signature category and sentence functor are provided. Several ways to introduce homomorphisms between models of pure first-order composition-nominative logic are proposed. Restrictions on signature morphisms necessary to construct model functor are proposed. Satisfaction condition for obtained constructions is shown to hold.

Keywords: Institutions, composition-nominative logics, model, irrefutability.

1 Introduction

Composition-nominative logics are program-oriented logics constructed according to principles of development from abstract to concrete, integrity of intensions and extensions, compositionality, and nominativity [1, 2, 3]. Distinctive feature of composition-nominative logics is high level of abstraction for data, which are not just elements of some set but rather carry additional structure of values assigned to names, i.e. they specify a correspondence between names and elements. This correspondence is called *nominative set* and can be partial. Depending on the entities studied the composition-nominative logics vary from propositional to program logics and form a hierarchy of such logics. Another dimension to differentiate one composition-nominative logic from another is whether its entities are interpreted as total or partial

functions. Properties of composition-nominative logics are quite well-studied [1, 3, 4]. Still there is a need to relate the results obtained for these logics to other logics. This can be achieved using such theoretical tools as institutions [5, 6].

Institutions are a unified framework that allows studying properties of logical systems in abstract manner independently of notation. Institutions capture a lot of common features of different logics. Institution-theoretic results are applicable to the wide range of logical systems. So considering the logical system one is interested in presenting it as institution and finding out what specificity the obtained institution has.

This paper initiates considerations on turning composition-nominative logics (CNLs) into institutions. To be informative pure first-order composition-nominative logic is selected. This logic is also known as pure quasiary predicate logics based on algebras with one sort. It is quite expressive and it is not too overloaded with details.

2 Pure first-order composition-nominative logic

2.1 Syntax

We start with the definition of syntax of the pure first-order CNL. Let P_s and V be the sets of predicate symbols and names respectively. The class of well-formed formulas is defined inductively (here we use notation similar to [7]):

$$\begin{array}{rcl} \Phi & ::= & \pi \\ & & \neg\Psi \\ & & \Psi \vee \Psi' \\ & & \exists x \Psi \\ & & R^{v_1 \ldots v_n}_{x_1 \ldots x_n} \Psi, \end{array} \tag{1}$$

where $\pi \in P_s$, $x, x_i, v_i \in V$; Ψ and Ψ' are formulas. Symbols $\neg$, $\vee$, $\exists x$, $R^{v_1 \ldots v_n}_{x_1 \ldots x_n}$ are called *composition* symbols. Composition $R^{v_1 \ldots v_n}_{x_1 \ldots x_n}$ is called *renomination* and usually abbreviated as $R^{\bar{v}}_{\bar{x}}$. It has uniqueness constraint on *upper* names v_i, i.e. $v_i = v_j$ only if $i = j$. Implication, conjunction and universal quantifier are defined conventionally as

follows

$$\Phi \wedge \Psi = \neg(\neg\Phi \vee \neg\Psi)$$
$$\Phi \to \Psi = \neg\Phi \vee \Psi$$
$$\forall x\Phi = \neg\exists x\neg\Phi$$

2.2 Nominative data

The basis for semantics of various composition-nominative logics is formed by nominative sets and quasiary predicates. Let $A \neq \emptyset$ be some set. A (*partial*) *nominative set* is a partial mapping from V to A, the class of all such mappings is denoted by ${}^{V}A$. In this context set A is called set of values, ${}^{V}A$ is called set of nominative sets or set of *states*. Nominative sets can be also called *nominative data*.

We use the following notation in regard to partiality. Let $f\colon A \xrightarrow{p} B$, $a \in A$, $b \in B$. We write $f(a)\uparrow$ if $a \notin \delta_f$, otherwise if $a \in \delta_f$ we write $f(a)\downarrow$. In the latter case $f(a)\downarrow$ can be used as well as the value of f on a, e.g. $f(a)\downarrow = b$.

The elements of nominative data are pairs of the form $v \mapsto a$. Expression $v \mapsto a \in_n d$ denotes $d(v)\downarrow = a$. Nominative sets are constructed using set-builder notation with square brackets.

Let us introduce the unary operation $r^{v_1\ldots v_n}_{x_1\ldots x_n}$ of finite renomination of nominative set. First, we specify a total function $\sigma^{v_1\ldots v_n}_{x_1\ldots x_n}\colon {}^{V}A \to {}^{V}A$ associated with it:

$$\sigma^{v_1\ldots v_n}_{x_1\ldots x_n}(v) = \begin{cases} x_i & \text{if } v = v_i. \\ v & \text{otherwise.} \end{cases}$$

Then $r^{v_1\ldots v_n}_{x_1\ldots x_n} d = d \circ \sigma^{v_1\ldots v_n}_{x_1\ldots x_n}$, where $\circ$ denotes partial functions composition.

We require another operation, single name overriding,

$$d\triangledown u \mapsto a = d\big|_{V-\{u\}} \dot\cup\, [u \mapsto a],$$

where $\big|_W$ denotes conventional restriction of function domain and $\dot\cup$ emphasizes that the union is disjoint.

Construction ${}^{V}A$ demonstrates bifunctorial behavior in the following way. Let $\sigma\colon V \to V'$, $f\colon A \to A'$ be two total functions. They induce several total functions between nominative set domains: function ${}^{\sigma}A\colon {}^{V'}A \to {}^{V}A$ that maps nominative set $d \in {}^{V'}A$ to nominative set $d \circ \sigma$, function ${}^{V}f\colon {}^{V}A \to {}^{V}A'$ that maps $d \in {}^{V}A$ to $f \circ d$, and function ${}^{\sigma}f\colon {}^{V'}A \to {}^{V}A'$ defined as $d \mapsto f \circ d \circ \sigma$. Notice that functions induced by change of set of values and set of names commute under composition:

$$ {}^{V}f \circ {}^{\sigma}A = {}^{\sigma}f = {}^{\sigma}A' \circ {}^{V'}f. \tag{2}$$

2.3 Quasiary predicates

Let $Bool = \{\top, \bot\}$ be a Boolean set. The *quasiary* predicate over set of names V is a partial Boolean-valued function: ${}^{V}A \xrightarrow{p} Bool$. Here letter p over arrow is used to emphasize the partiality of this mapping. The quasiary predicates over set of names V are called V-quasiary predicates for short. Let $Pr^{V}_{A} = \left\{ p \mid p\colon {}^{V}A \xrightarrow{p} Bool \right\}$.

The truth and falsity domains of $p \in Pr^{V}_{A}$ are respectively $T(p) = \{d \mid p(d)\downarrow = \top\} = p^{-1}(\{\top\})$, $F(p) = p^{-1}(\{\bot\})$.

Definition 1. *An* extension *of a partial predicate p is a pair of its truth and falsity domains:* $\|p\| = (T(p), F(p))$.

Notice that sets in extension of a predicate should be disjoint. There is a 1-1 correspondence between extensions and partial predicates. Also there is a natural ordering of extensions:

$$\|p\| \subseteq \|p'\| \text{ if } T(p) \subseteq T(p') \text{ and } F(p') \subseteq F(p).$$

Definition 2. *A predicate p is* irrefutable *if $F(p) = \emptyset$.*

Similarly to domain of nominative data ${}^{V}A$ construction Pr^{V}_{A} also has bifunctorial behavior. Given two total maps $\sigma\colon V \to V'$, $f\colon A \to A'$ there are total maps $Pr^{V}_{f}\colon Pr^{V}_{A'} \to Pr^{V}_{A}$, $Pr^{\sigma}_{A}\colon Pr^{V}_{A} \to Pr^{V'}_{A}$,

$Pr_f^\sigma \colon Pr_{A'}^V \to Pr_A^{V'}$ realized as follows. Let $p \in Pr_{A'}^V$, $q \in Pr_A^V$, then

$$Pr_f^V(p) = p \circ {}^V f$$
$$Pr_A^\sigma(q) = q \circ {}^\sigma A$$
$$Pr_f^\sigma(p) = p \circ {}^\sigma f.$$

Once again notice that maps induced by change of set of values and set of names commute under composition

$$Pr_f^{V'} \circ Pr_{A'}^\sigma = Pr_f^\sigma = Pr_A^\sigma \circ Pr_f^V. \tag{3}$$

2.4 Semantics

The set Pr_A^V is used as a carrier set for most composition-nominative logics. Compositions have fixed interpretation for CNLs and are defined as follows

$$\begin{aligned}
\|p \vee q\| &= (T(p) \cup T(q), F(p) \cap F(q)) \\
\|\neg p\| &= (F(p), T(p)) \\
\|\exists x p\| &= (\{d \mid d\nabla x \mapsto a \in T(p) \text{ for some } a \in A\}, \\
&\qquad \{d \mid d\nabla x \mapsto a \in F(p) \text{ for all } a \in A\}) \\
(R_{\bar{x}}^{\bar{v}} p)(d) &= p(r_{\bar{x}}^{\bar{v}} d) = p(d \circ \sigma_{\bar{x}}^{\bar{v}}) = p({}^{\sigma_{\bar{x}}^{\bar{v}}} A(d)) = Pr_A^{\sigma_{\bar{x}}^{\bar{v}}}(p)(d).
\end{aligned}$$

Definition 3. *A* first-order algebra of pure V-quasiary predicates *is a tuple* $(Pr, A; \vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x)$ *where set* $Pr \subseteq Pr_A^V$ *is closed under compositions.*

Definition 4. *Given a set of names* V *and predicate symbols* P_s *a* model *of first-order pure quasiary predicates logic is a triple* (Pr, A, I) *such that* $(Pr, A; \vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x)$ *is a pure* V*-quasiary predicates algebra, and* $I \colon P_s \to Pr$ *is a total mapping.*

3 Construction of institution

3.1 Outline of adaptation process

Recall the definition of the institution [5, p.27-28].

Definition 5. *An* institution *is a tuple* $I = (\mathbb{S}ig, \mathrm{Sen}, \mathrm{Mod}, \models)$ *where*

- $\mathbb{S}ig$ *is a category of signatures;*
- $\mathrm{Sen}\colon \mathbb{S}ig \to \mathbb{S}et$ *is a "sentence" functor;*
- $\mathrm{Mod}\colon \mathbb{S}ig \to \mathbb{C}at^{\mathrm{op}}$ *is a contravariant model functor;*
- $\models \subseteq |\mathrm{Mod}(\Sigma)| \times \mathrm{Sen}(\Sigma)$ *for each* $\Sigma \in |\mathbb{S}ig|$ *is a satisfaction relation;*

such that the following holds for any signature morphism $\varphi\colon \Sigma \to \Sigma'$ *and sentence* $e \in \mathrm{Sen}(\Sigma)$*:*

$$M' \models \mathrm{Sen}(\varphi)(e) \text{ if and only if } \mathrm{Mod}(\varphi)(M') \models e.$$

This property is called satisfaction condition.

Knowing the ingredients of institution we can propose steps required to adopt logic in question as institution:

1. Select signatures and define signature morphisms.
2. Extend signature morphisms to sentence translation maps.
3. Define morphism between models and extend Mod to functor.
4. Provide satisfaction relation and check satisfaction condition.

From these first two steps are straight-forward, the last two are quite challenging. Next we present considerations on each of the step for pure first-order CNL.

3.2 Signature morphisms and sentence translation

Generally speaking signature in first-order pure quasiary predicate logic is a tuple $(V, P_s, \vee, \neg, R^{\bar{v}}_{\bar{x}}, \exists x)$. However, since part $\vee, \neg, R^{\bar{v}}_{\bar{x}}, \exists x$ is fixed and present in any language of pure first-order CNL, we can confine ourselves to only first two components. Thus signature in first-order pure quasiary predicate logic is $\Sigma_Q = (V, P_s)$, where V is a set of names, P_s set of predicate symbols.

Definition 6. *A* morphism *of signatures is* $\varphi = (\varphi_V, \varphi_P)\colon (V, P_s) \to (V', P'_s)$*, where* $\varphi_V\colon V \to V'$ *is a bijection and* $\varphi_P\colon P_s \to P'_s$*.*

Name component φ_V of signature morphism is restricted to bijections for several reason. To avoid name clashes in renomination composition it has to be 1-1 mapping. Another need to extend Mod to a functor forces surjectivity requirement on this mapping.

Our category $\mathbb{S}ig$ is simply a category of signatures and signature morphisms.

Now we can extend action of signature morphism to the Σ_Q-sentences defined in (1), i.e. define $\mathrm{Sen}(\varphi)\colon \mathrm{Sen}(V, P_s) \to \mathrm{Sen}(V', P'_s)$. This is done by induction on structure of the sentence.

$$\begin{aligned}
\mathrm{Sen}(\varphi)(\pi) &= \varphi_P(\pi)\\
\mathrm{Sen}(\varphi)(\Phi \vee \Psi) &= \mathrm{Sen}(\varphi)(\Phi) \vee \mathrm{Sen}(\varphi)(\Psi)\\
\mathrm{Sen}(\varphi)(\neg\Phi) &= \neg\,\mathrm{Sen}(\varphi)(\Phi)\\
\mathrm{Sen}(\varphi)(\exists x\Phi) &= \exists\varphi_V(x)\,\mathrm{Sen}(\varphi)(\Phi)\\
\mathrm{Sen}(\varphi)(R^{\bar v}_{\bar x}\Phi) &= R^{\varphi_V(\bar v)}_{\varphi_V(\bar x)}\,\mathrm{Sen}(\varphi)(\Phi).
\end{aligned}$$

Lemma 1. $\mathbb{S}ig$ *is a category.* Sen *is a functor* $\mathbb{S}ig \to \mathbb{S}et$*.*

In a context where Sen is known expression $\mathrm{Sen}(\varphi)(\Phi)$ is usually abbreviated as simply $\varphi(\Phi)$.

3.3 Homomorphisms between models

Consider conventional case of pure first-order logic. Model homomorphisms are functions $f\colon A \to B$ with predicate preservation property. For each arity n due to contravariant powerset functor there is an induced map $\mathcal{P}_n(f)\colon \mathcal{P}(B^n) \to \mathcal{P}(A^n)$ between n-ary predicates. Preservation of n-ary predicate symbol $\pi \in P_n$ means $M_\pi \subseteq \mathcal{P}_n(f)(M'_\pi)$.

An analogous construction for quasiary case is presented in subsection 2.3. Let $f\colon A \to A'$ be a total map. Consider total map $Pr^V_f\colon Pr^V_{A'} \to Pr^V_A$ induced by f. It has some attractive properties.

Lemma 2. *Function Pr^V_f preserves disjunction, negation and renomination compositions. If f is surjective, it also preserves existential quantifier composition.*

Proof. Let $p \in Pr^V_{A'}$, then

$$T(Pr^V_f(p)) = \{d \mid {}^Vf(d) \in T(p)\} = \left({}^Vf\right)^{-1}(T(p)).$$

Therefore

$$\begin{aligned}\|Pr^V_f(p)\| &= (T(Pr^V_f(p)), F(Pr^V_f(p))) = (\left({}^Vf\right)^{-1}(T(p)), \left({}^Vf\right)^{-1}(F(p))) \\ &= \left({}^Vf\right)^{-1}\|p\|.\end{aligned}$$

Let $p, q \in Pr^V_{A'}$, then

$$\begin{aligned}\|Pr^V_f(\neg p)\| &= \left({}^Vf\right)^{-1}(F(p), T(p)) = \|\neg Pr^V_f(p)\| \\ \|Pr^V_f(p \vee q)\| &= \left({}^Vf\right)^{-1}(T(p) \cup T(q), F(p) \cap F(q)) \\ &= \|Pr^V_f(p) \vee Pr^V_f(q)\|,\end{aligned}$$

where preservation of unions and intersections by preimage is used.

For renomination composition we use commutativity (3):

$$Pr^V_f(R^{\bar{v}}_{\bar{x}}p) = Pr^V_f \circ Pr^{\sigma^{\bar{v}}_{\bar{x}}}_{A'}(p) = Pr^{\sigma^{\bar{v}}_{\bar{x}}}_{A} \circ Pr^V_f(p) = R^{\bar{v}}_{\bar{x}}Pr^V_f(p).$$

Finally, if $f\colon A \to A'$ is surjective, then

$$\begin{aligned}T(Pr^V_f(\exists x p)) &= \left\{d \mid (f \circ d)\triangledown x \mapsto a' \in T(p) \text{ for some } a' \in A'\right\} \\ &= \{d \mid (f \circ d)\triangledown x \mapsto f(a) \in T(p) \text{ for some } a \in A\} \\ &= \{d \mid f \circ (d\triangledown x \mapsto a) \in T(p) \text{ for some } a \in A\}) \\ &= T(\exists x Pr^V_f(p)). \\ F(Pr^V_f(\exists x p)) &= \left\{d \mid (f \circ d)\triangledown x \mapsto a' \in F(p) \text{ for all } a' \in A'\right\} \\ &= \{d \mid (f \circ d)\triangledown x \mapsto f(a) \in F(p) \text{ for all } a \in A\} \\ &= F(\exists x Pr^V_f(p)).\end{aligned}$$

That is $Pr^V_f(\exists x p) = \exists x Pr^V_f(p)$. □

Thus we only need to formalize preservation of predicates by map f. There are several ways to accomplish this. Here we do it closest to conventional case using the extensions of quasiary predicate.

Definition 7. *A* (V, P_s)-model homomorphism $f\colon (Pr, A, I) \to (Pr', A', I')$ *is a total map* $f\colon A \to A'$ *such that* $Pr^V_f(Pr') \subseteq Pr$ *and* $\|I_\pi\| \subseteq \|Pr^V_f(I'_\pi)\|$ *for all* $\pi \in P_s$.

Lemma 3. (V, P_s)*-models and* (V, P_s)*-model homomorphisms form a category* $|\mathrm{Mod}(V, P_s)|$.

If this notion of homomorphism is too strict other options include different relations between extensions or use of implication. Notice that immediate usage of implication composition is not an option because implication is not transitive w.r.t. irrefutability. So usage of implication involves its modification or departing from irrefutability as a criteria for predicate.

The homomorphisms between models can be generalized as maps $Pr^V_{A'} \to Pr^V_A$ that have preservation properties for compositions and preserve predicates according to chosen rule.

3.4 Reduct functor

Recall that name component of signature morphism is a bijection $\varphi_V\colon V \to V'$. It induces bijections ${}^{\varphi_V}A\colon {}^{V'}A \to {}^{V}A$, $Pr^{\varphi_V}_A\colon Pr^V_A \to Pr^{V'}_A$. This was needed to be able to jump from $Pr^{V'}_A$ to Pr^V_A as Mod-functor implies. Before working out change of the model let us see how φ_V affects the extensions of quasiary predicate and how it interacts with renomination.

Lemma 4. $q = Pr^{\varphi_V^{-1}}_A(p)$ *if and only if* $\|q\| = {}^{\varphi_V}A(\|p\|)$, *i.e. extension of* q *equals to pairwise application of* ${}^{\varphi_V}A$ *to extension of* p.

Proof. Let $p \in Pr^{V'}_A$, $q \in Pr^V_A$ such that $q = Pr^{\varphi_V^{-1}}_A(p)$, where φ_V^{-1} is inverse of φ_V. Then

$$\begin{aligned} T(q) &= \{d \in {}^{V}A \mid d \circ \varphi_V^{-1} \in T(p)\} \\ &= \{d' \circ \varphi_V \mid d' \in T(p)\} = {}^{\varphi_V}A(T(p)). \end{aligned}$$

By analogy $F(q) = {}^{\varphi_V}A(F(p))$. Since partial predicate is fully determined by its extension we conclude the statement of the lemma. □

Lemma 5. *Following diagram commutes*

$$\begin{array}{ccc} V & \xrightarrow{\sigma^{\bar{v}}_{\bar{x}}} & V \\ {\scriptstyle \varphi_V}\downarrow & & \downarrow{\scriptstyle \varphi_V} \\ V' & \xrightarrow[\sigma^{\varphi_V(\bar{v})}_{\varphi_V(\bar{x})}]{} & V' \end{array}$$

Next we provide a model transformation.

Lemma 6. *Let $\varphi\colon (V, P_s) \to (V', P'_s)$ be a signature morphism and (Pr', A, I') be a (V', P'_s)-model. Then triple $(Pr_A^{\varphi_V^{-1}}(Pr'), A, Pr_A^{\varphi_V^{-1}} \circ I' \circ \varphi_P)$ is a (V, P_s)-model.*

Proof. Suppose that $q_{1,2} \in Pr_A^{\varphi_V^{-1}}(Pr')$. Then there are $p_{1,2} \in Pr'$ such that $\|q_i\| = {}^{\varphi_V}A(\|p_i\|)$. Then

$$\begin{aligned} \|q_1 \vee q_2\| &= (T(q_1) \cup T(q_2), F(q_1) \cap F(q_2)) \\ &= ({}^{\varphi_V}A(T(p_1)) \cup {}^{\varphi_V}A(T(p_2)), {}^{\varphi_V}A(F(p_1)) \cap {}^{\varphi_V}A(F(p_2))) \\ &= ({}^{\varphi_V}A(T(p_1) \cup T(p_2)), {}^{\varphi_V}A(F(p_1) \cap F(p_2))) \\ &= {}^{\varphi_V}A(\|p_1 \vee p_2\|). \end{aligned}$$

Here we used properties of the image of bijection.

By analogy for $q \in Pr_A^{\varphi_V^{-1}}(Pr')$ we have

$$\|\neg q\| = (F(q), T(q)) = {}^{\varphi_V}A(F(p), T(p)) = {}^{\varphi_V}A(\|\neg p\|).$$

For existential quantifier

$$\begin{aligned} T(\exists x Pr_A^{\varphi_V^{-1}}(p)) &= \{d \mid (d \triangledown x \mapsto a) \circ \varphi_V^{-1} \in T(p) \text{ for some } a\} \\ &= \{d \mid d \circ \varphi_V^{-1} \triangledown \varphi_V(x) \mapsto a \in T(p) \text{ for some } a\} \\ &= T(Pr_A^{\varphi_V^{-1}}(\exists \varphi_V(x) p)). \end{aligned}$$

Repeating for falsity domain and combining we derive

$$\exists x Pr_A^{\varphi_V^{-1}}(p) = Pr_A^{\varphi_V^{-1}}(\exists \varphi_V(x) p).$$

For renomination by lemma 5 we immediately have

$$R_{\bar{x}}^{\bar{v}} Pr_A^{\varphi_V^{-1}}(p) = Pr^{\sigma_{\bar{x}}^{\bar{v}}} \circ Pr_A^{\varphi_V^{-1}}(p) = Pr_A^{\varphi_V^{-1}}(R_{\varphi_V(\bar{x})}^{\varphi_V(\bar{v})} p).$$

Considerations above show that $Pr_A^{\varphi_V^{-1}}(Pr')$ is closed under quasiary predicates compositions. Thus $(Pr_A^{\varphi_V^{-1}}(Pr'), A; \vee, \neg, R_{\bar{v}}^{\bar{x}}, \exists x)$ is a V-quasiary algebra.

Now notice that we have following diagram

$$P_s \xrightarrow{\varphi_P} P_s' \xrightarrow{I'} Pr' \xrightarrow{Pr_A^{\varphi_V^{-1}}} Pr_A^{\varphi_V^{-1}}(Pr')$$

which means that $Pr_A^{\varphi_V^{-1}} \circ I' \circ \varphi_P \colon P_s \to Pr_A^{\varphi_V^{-1}}(Pr')$ and finishes the proof. □

Lemma 7. *Given a signature morphism* $\varphi \colon (V, P_s) \to (V', P_s')$ *any* (V', P_s')*-model homomorphism* $f \colon (Pr', A, I') \to (Pr_1', A_1, I_1')$ *is also* (V, P_s)*-model homomorphism* $f \colon (Pr_A^{\varphi_V^{-1}}(Pr'), A, Pr_{A'}^{\varphi_V^{-1}} \circ I' \circ \varphi_P) \to (Pr_{A_1}^{\varphi_V^{-1}}(Pr_1'), A_1, Pr_{A_1}^{\varphi_V^{-1}} \circ I_1' \circ \varphi_P)$.

Proof. Proof is based on the fact that maps $Pr_{A'}^{\varphi_V^{-1}}$ and Pr_f^V commute. Preservation of predicates by resulting (V, P_s)-model homomorphism is fully reduced to preservation of predicates by original (V', P_s')-model homomorphism. □

Corollary 8. *Construction*

$$\begin{aligned} \mathrm{Mod}(\varphi)(Pr', A', I') &= (Pr_{A'}^{\varphi_V^{-1}}(Pr'), A', Pr_{A'}^{\varphi_V^{-1}} \circ I' \circ \varphi_P) \\ \mathrm{Mod}(\varphi)(f) &= f \end{aligned}$$

extends Mod *to a functor* $\mathbb{S}ig \to \mathbb{C}at$.

When Mod is known the model resulting from application of reduct functor $\mathrm{Mod}(\varphi)$ to M, i.e. $\mathrm{Mod}(\varphi)(M)$, is often abbreviated as $M\big|_\varphi$.

3.5 Satisfaction relation

First we extend interpretation to all formulas. Due to definition of quasiary predicate algebras it is quite easy. Let $\Phi \in \mathrm{Sen}(V, P_s)$, $M = (Pr, A, I) \in |\mathrm{Mod}(V, P_s)|$. We define $M(\Phi) \in Pr$ inductively:

$$\begin{aligned} M(\pi) &= I(\pi) \\ M(\Phi \vee \Psi) &= M(\Phi) \vee M(\Psi) \\ M(\neg\Phi) &= \neg M(\Phi) \\ M(\exists x \Phi) &= \exists x M(\Phi) \\ M(R^{\bar{v}}_{\bar{x}}\Phi) &= R^{\bar{v}}_{\bar{x}} M(\Phi) \end{aligned}$$

In the right-hand side we use interpretation of composition symbols given in subsection 2.4.

Definition 8. *Formula $\Phi \in \mathrm{Sen}(V, P_s)$ is* satisfied *by (V, P_s)-model $M = (Pr, A, I) \in |\mathrm{Mod}(V, P_s)|$, if predicate $M(\Phi)$ is irrefutable, i.e. $F(M(\Phi)) = \emptyset$. This is denoted by $M \models \Phi$.*

Let us see how change of notation affects interpretation of a formula.

Lemma 9. *Given formula $\Phi \in \mathrm{Sen}(V, P_s)$, (V', P'_s)-model $M' = (Pr', A, I')$ and signature morphism $\varphi\colon (V, P_s) \to (V', P'_s)$, the following holds:*

$$Pr_A^{\varphi_V^{-1}}(M'(\varphi(\Phi))) = M'\big|_\varphi(\Phi).$$

Proof. By induction on structure of formula Φ. Let $\pi \in P_s$ be a predicate symbol, then

$$\begin{aligned} Pr_A^{\varphi_V^{-1}}(M'(\varphi_P(\pi))) &= Pr_A^{\varphi_V^{-1}}(I'(\varphi_P(\pi))) = Pr_A^{\varphi_V^{-1}} \circ I' \circ \varphi_P(\pi) \\ &= M'\big|_\varphi(\pi). \end{aligned}$$

For the following cases we use the argument of the proof of lemma 6 and induction hypothesis:

$$\begin{aligned} Pr_A^{\varphi_V^{-1}}(M'(\varphi(\Phi \vee \Psi))) &= Pr_A^{\varphi_V^{-1}}\left(M'(\varphi(\Phi)) \vee M'(\varphi(\Psi))\right) \\ &= Pr_A^{\varphi_V^{-1}}\left(M'(\varphi(\Phi))\right) \vee Pr_A^{\varphi_V^{-1}}\left(M'(\varphi(\Psi))\right) \\ &= M'\big|_\varphi(\Phi) \vee M'\big|_\varphi(\Psi) = M'\big|_\varphi(\Phi \vee \Psi) \end{aligned}$$

$$
\begin{aligned}
Pr_A^{\varphi_V^{-1}}(M'(\varphi(\neg\Phi))) &= \neg Pr_A^{\varphi_V^{-1}}(M'(\varphi(\Phi))) \\
&= \neg M'\big|_\varphi(\Phi) = M'\big|_\varphi(\neg\Phi) \\
Pr_A^{\varphi_V^{-1}}(M'(\varphi(\exists x\Phi))) &= Pr_A^{\varphi_V^{-1}}(\exists\varphi_V(x)M'(\varphi(\Phi))) \\
&= \exists x Pr_A^{\varphi_V^{-1}}(M'(\varphi(\Phi))) = M'\big|_\varphi(\exists x\Phi) \\
Pr_A^{\varphi_V^{-1}}(M'(\varphi(R_{\bar{x}}^{\bar{v}}\Phi))) &= Pr_A^{\varphi_V^{-1}}(R_{\varphi_V(\bar{x})}^{\varphi_V(\bar{v})}M'(\varphi(\Phi))) \\
&= R_{\bar{x}}^{\bar{v}} Pr_A^{\varphi_V^{-1}}(M'(\varphi(\Phi))) = M'\big|_\varphi(R_{\bar{x}}^{\bar{v}}\Phi). \qquad \square
\end{aligned}
$$

Corollary 10. *Given formula* $\Phi \in \mathrm{Sen}(V, P_s)$, (V', P'_s)*-model* $M' = (Pr', A, I')$ *and signature morphism* $\varphi\colon (V, P_s) \to (V', P'_s)$, *the following holds:*

$$M' \models \varphi(\Phi) \text{ if and only if } M'\big|_\varphi \models \Phi.$$

Proof. By previous lemma $F(M'\big|_\varphi(\Phi)) = {}^{\varphi_V}A(F(M'(\varphi(\Phi)))$. Due to properties of images the satisfaction condition holds. $\square$

By lemma 1, and corollaries 8, 10 we have

Theorem 1. *Constructed* $(\mathbb{S}ig, \mathrm{Sen}, \mathrm{Mod}, \models)$ *form an institution.*

This result presents a pure first-order quasiary predicate logic as institution.

4 Conclusion

In this paper we propose how pure first-order quasiary predicate logic can be represented as institution. For this all necessary constituents of institution are provided. We consider possible alternative definitions of homomorphisms between models of pure first-order CNL. This construction can be further developed to cover other representatives of CNL family. Other line of research suggests studying the distinctive features of obtained institutions compared to more conventional ones.

References

[1] M. Nikitchenko, S. Shkilniak. *Mathematical logic and theory of algorithms*. Publishing house of Taras Shevchenko National University of Kyiv, Kyiv, 2008. (In Ukrainian)

[2] M. Nikitchenko, A. Chentsov. *Basics of Intensionalized Data: Presets, Sets, and Nominats*. Computer Science Journal of Moldova, vol. **20**, no. 3(60), 2012, pp. 334–365.

[3] M. Nikitchenko, V. Tymofieiev. *Satisfiability in composition-nominative logics*. Central European Journal of Computer Science, vol. **2**, no. 3 (2012), pp. 194–213.

[4] A. Kryvolap, M. Nikitchenko, W. Schreiner. *Extending Floyd-Hoare logic for partial pre- and postconditions*. CCIS, vol. **412** (2013), pp. 355–378.

[5] R. Diaconescu. *Institution-independent Model Theory*. Birkhäuser Basel, 2008.

[6] D. Sannella, A. Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. Springer, 2012.

[7] B. Pierce. *Types and Programming Languages*. MIT Press, 2002.

Alexey Chentsov, Mykola Nikitchenko, Received July 12, 2015

Alexey Chentsov
Taras Shevchenko National University of Kyiv
01601, Kyiv, Volodymyrska st, 60
Phone: +38044 2590511
E–mail: chentsov@ukr.net

Mykola Nikitchenko
Taras Shevchenko National University of Kyiv
01601, Kyiv, Volodymyrska st, 60
Phone: +38044 2590519
E–mail: nikitchenko@unicyb.kiev.ua

Part 2

Theoretical aspects of software system development

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Analysis of Nominative Data Sets Structure

Volodymyr G. Skobelev, Ievgen Ivanov, Mykola Nikitchenko

Abstract

The paper deals with several mathematical problems associated with a mathematical model of data in computing systems (nominative sets) used in the composition-nominative approach to software system formalization. In this paper the structure of the partially-ordered set of nominative sets is investigated in terms of set theory, lattice theory and algebraic systems theory. To achieve this aim the correct transferring of basic set-theoretic operations to nominative sets is investigated in detail. A basic partially ordered algebraic system intended to provide correct analogues of basic set-theoretic operations for nominative sets is elaborated and its structure is investigated. It is established that this structure is a lower semilattice. Properties of lower and upper cones of subsets of the poset of nominative sets are investigated in detail. Subsets for which upper cones are non-empty are characterized. Closed intervals of nominative sets are examined. Boolean algebra is determined on any such interval. A criterion for isomorphism of two closed intervals is obtained. Maximal closed intervals are investigated. It is established that the poset of nominative sets is a union of isomorphic overlapping maximal closed intervals.

Keywords: nominative set, nominative data, set theory, lattice theory, algebraic system, lower semilattice, lower and upper cones, closed intervals.

1 Introduction

Composition-nominative approach [1] aims to give a mathematical basis for development of formal methods of analysis of software systems. It is based on several principles, including the *Development principle*

(from abstract to concrete), *Principle of priority of semantics over syntax*, *Compositionality principle*, and *Nominativity principle*. The latter Nominativity principle states that so called nominative data are adequate models of data processed and stored in computing systems. Nominative data are based on naming relations that associate names and values. The simplest type of nominative data is a nominative set (or named set) which is a partial function from a set of names to a class values. It can be considered as a generalization of the notion of a program state considered as a total assignment of values to variable names. Other types of nominative data represent hierarchical data organizations [1]. These types of data can represent various data structures used in programming (e.g. multidimensional arrays, records, lists, trees, etc.). Other ideas of composition-nominative approach are that partial functions over nominative data represent semantics of programs (operating on data). Such functions are constructed from basic functions using operations called compositions which represent semantics of programming language constructs (e.g. sequential composition, branching, cycle, etc.). A set of programs (modeled as functions over nominative data) together with compositions forms an algebraic structure representing compositional semantics of a language. Thus any logical theory based on the composition-nominative approach is, in essence, some formal theory of partial mappings of a special type. Various associated logics [1] allow reasoning about semantic properties of programs.

The relevance of the composition-nominative approach increases significantly in connection with intensive investigation of SMT-solvers (i.e. intended for testing satisfiability of formulas of 1^{st}-order theories) [7]. From this viewpoint composition-nominative approach represents essential development of the theory of equality and uninterpreted functions, the foundations of which were laid in [8].

Although applications are important, composition-nominative approach is largely an algebraic framework, so it gives a rise to certain theoretical questions. One of such questions is investigation of nominative sets from the perspective of abstract algebra with the perspective of applications of the discovered algebraic properties of mathematical

models of program data in automated software analysis and synthesis. This is the topic of this paper.

The main goal of this paper is analysis of the structure of arbitrary nominative data sets in terms of partial-ordered sets theory [9], lattice theory [10] and algebraic systems theory [11].

2 Basic notions

Let V and A be non-empty finite or countable sets of names and data respectively. The set $\mathfrak{F}_{V,A}$ of all V-nominative sets over A is the set of all (possibly, partial) mappings from V to A.

If $|A| = 1$, then $\mathfrak{F}_{V,A}$ can be considered as the set $\mathcal{B}(V)$ of all subsets of the set V, while if $|V| = 1$, then $\mathfrak{F}_{V,A}$ can be considered as the set consisting of the empty set and all 1-element subsets of the set A. Thus in what follows it is supposed that $|V| \geq 2$ and $|A| \geq 2$.

We will deal with the set $\mathfrak{G}_{V,A} = \{graph(f) | f \in \mathfrak{F}_{V,A}\}$, where $graph(f) = \{(v, a) \in \mathrm{Dom}f \times \mathrm{Val}f | f(v) = a\}$.

The following partial ordering can be defined on the set $\mathfrak{F}_{V,A}$:

$$f_1 \preceq f_2 \Leftrightarrow graph(f_1) \subseteq graph(f_2) \ \ (f_1, f_2 \in \mathfrak{F}_{V,A}). \qquad (1)$$

The least element of the poset $\mathfrak{F}_{V,A}$ is the V-nominative set $0_{V,A}$ with empty domain, while the set of all maximal elements of the poset $\mathfrak{F}_{V,A}$ is the set $\mathfrak{F}^{(ttl)}_{V,A}$ of all total V-nominative sets.

Since $|A| \geq 2$, for any set of names V ($|V| \geq 2$) the poset $(\mathfrak{F}_{V,A}, \preceq)$ does not have the largest element. Thus this poset is not isomorphic to any Boolean algebra.

We write $f_1 \prec f_2$ ($f_1, f_2 \in \mathfrak{F}_{V,A}$) if and only if $f_1 \preceq f_2$ and $f_1 \neq f_2$. By "$\succeq$" (or, respectively, by "$\succ$") we denote the relation that is an inverse of the relation " $\preceq$ " (or, respectively, of the relation "$\prec$").

3 Algebra of nominative sets

In this section we will transfer the basic set-theoretic operations to the set $\mathfrak{F}_{V,A}$ with the purpose of providing correct operations on V-

nominative sets over A with perspectives of application in automation of software development and analysis.

The unary set-theoretic operation of complement of a set cannot be transferred to $\mathfrak{F}_{V,A}$ since this set does not contain the largest element.

Let us transfer binary set-theoretic operations to the set $\mathfrak{F}_{V,A}$.

There are no difficulties with transferring the set-theoretic operations of intersection of two sets "$\cap$" and the difference of two sets "$\setminus$" to the set $\mathfrak{F}_{V,A}$. Indeed, for any $f_1, f_2, f \in \mathfrak{F}_{V,A}$ we can define:

$$f_1 \cap f_2 = f \Leftrightarrow graph(f_1) \cap graph(f_2) = graph(f), \qquad (2)$$
$$f_1 \setminus f_2 = f \Leftrightarrow graph(f_1) \setminus graph(f_2) = graph(f).$$

Obviously, for any $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following formulas hold: $\mathrm{Dom}(f_1 \cap f_2) \subseteq \mathrm{Dom}f_1 \cap \mathrm{Dom}f_2$, $f_1|_X \cap f_2|_Y = (f_1 \cap f_2)|_{X \cap Y}$ $(X, Y \subseteq V)$ and $\mathrm{Dom}f_1 \setminus \mathrm{Dom}f_2 \subseteq \mathrm{Dom}(f_1 \setminus f_2) \subseteq \mathrm{Dom}f_1$. The following two propositions are true:

Proposition 1. *Algebraic system $(\mathfrak{F}_{V,A}, \cap)$ is a commutative semigroup without neutral element, but with zero element which is the V-nominative set $0_{V,A}$ with empty domain.*

Proposition 2. *Algebraic system $(\mathfrak{F}_{V,A}, \setminus)$ is a non-commutative non-associative magma in which the V-nominative set $0_{V,A}$ with the empty domain is both the right identity element and the left zero element.*

A different situation occurs with transferring of the operations of the union of two sets "$\cup$" and of the symmetric difference of two sets "$\oplus$" to the set $\mathfrak{F}_{V,A}$. Indeed, for any $f_1, f_2 \in \mathfrak{F}_{V,A}$ we get:

$$graph(f_1) \cup graph(f_2) \in \mathfrak{G}_{V,A} \Leftrightarrow f_1|_{\mathrm{Dom}f_1 \cap \mathrm{Dom}f_2} = f_2|_{\mathrm{Dom}f_1 \cap \mathrm{Dom}f_2},$$

$$graph(f_1) \oplus graph(f_2) \in \mathfrak{G}_{V,A} \Leftrightarrow f_1|_{\mathrm{Dom}f_1 \cap \mathrm{Dom}f_2} = f_2|_{\mathrm{Dom}f_1 \cap \mathrm{Dom}f_2}.$$

Thus the formulas

$$f_1 \cup f_2 = f \Leftrightarrow graph(f_1) \cup graph(f_2) = graph(f) \ \ (f_1, f_2, f \in \mathfrak{F}_{V,A}),$$

$$f_1 \oplus f_2 = f \Leftrightarrow graph(f_1) \oplus graph(f_2) = graph(f) \ \ (f_1, f_2, f \in \mathfrak{F}_{V,A})$$

can define only partial operations on the set $\mathfrak{F}_{V,A}$.

In order to avoid such a situation we transfer the operations "$\cup$" and "$\oplus$" to the set $\mathfrak{F}_{V,A}$ as follows: for any $f_1, f_2, f \in \mathfrak{F}_{V,A}$ we define:

$$f_1 \triangleright f_2 = f \Leftrightarrow graph(f_1) \cup graph(f_2|_{\mathrm{Dom}f_2 \setminus \mathrm{Dom}f_1}) = graph(f)$$

and

$$f_1 \boxplus f_2 = f \Leftrightarrow$$

$$\Leftrightarrow graph(f_1|_{\mathrm{Dom}f_1 \setminus \mathrm{Dom}f_2}) \cup graph(f_2|_{\mathrm{Dom}f_2 \setminus \mathrm{Dom}f_1}) = graph(f).$$

It is worth noting that these two operations are intended to join together any two V-nominative sets over A. The operation " $\triangleright$ " is called overlapping (of the second nominative set by the first one), the operation " $\boxplus$ " can be called the exclusive compound. The following proposition is true:

Proposition 3. *For any $f_1, f_2, f_3 \in \mathfrak{F}_{V,A}$ the following formulas hold:*
(i) $\mathrm{Dom}(f_1 \triangleright f_2) = \mathrm{Dom}f_1 \cup \mathrm{Dom}f_2$;
(ii) $f_1 \preceq f_1 \triangleright f_2$;
(iii) $f_1 \preceq f_2 \Rightarrow f_1 \triangleright f_2 = f_2 \triangleright f_1 = f_2$;
(iv) $(f_1 \triangleright f_2) \cap f_3 \preceq (f_1 \cap f_6) \triangleright (f_2 \cap f_3)$;
(v) $f_3 \cap (f_1 \triangleright f_2) \preceq (f_1 \cap f_3) \triangleright (f_2 \cap f_3)$;
(vi) $(f_1 \cap f_2) \triangleright f_3 \succeq (f_1 \triangleright f_3) \cap (f_2 \triangleright f_3)$;
(vii) $f_1 \triangleright (f_2 \cap f_3) = (f_1 \triangleright f_2) \cap (f_1 \triangleright f_3)$.

It is not difficult to give examples showing that there may be strict inequalities in the formulas (ii), (iv)-(vi).

The following theorem is true:

Theorem 1. *The algebraic system $(\mathfrak{F}_{V,A}, \triangleright)$ is a non-commutative monoid with neutral element which is the V-nominative set $0_{V,A}$ with the empty domain.*

Proposition 1 and Theorem 1 imply that the algebraic system $(\mathfrak{F}_{V,A}, \triangleright, \cap)$ differs from well-known algebraic systems with two binary operations (i.e. a field, a ring, a semi-ring, etc.). Thus the properties of

the set of all valid formulas in the algebraic system $(\mathfrak{F}_{V,A}, \triangleright, \cap)$ can substantially differ from the properties of the sets of all valid formulas in standard algebraic systems with two binary operations. The following proposition is true:

Proposition 4. *The algebraic system* $(\mathfrak{F}_{V,A}, \boxplus)$ *is a commutative semigroup with the neutral element which is the* V*-nominative set* $0_{V,A}$ *with empty domain.*

Since $(f_1 \boxplus f_2) \cap f_3 = (f_1 \cap f_3) \boxplus (f_2 \cap f_3)$ for any $f_1, f_2, f_3 \in \mathfrak{F}_{V,A}$, Propositions 1 and 4 imply that the following theorem is true

Theorem 2. *The algebraic system* $(\mathfrak{F}_{V,A}, \cap, \boxplus)$ *is a semiring.*

Thus we have defined an algebraic system $(\mathfrak{F}_{V,A}, \mathfrak{O}_{V,A}, \mathfrak{R}_{V,A})$, where $\mathfrak{F}_{V,A}$ is the base, $\mathfrak{O}_{V,A} = \{\cap, \setminus, \triangleright, \boxplus\}$ is the set of operations and $\mathfrak{R}_{V,A} = \{=, \preceq\}$ is the set of relations.

It is worth noting that since the operations $\cap$ and $\boxplus$ are associative and can be naturally extended to any finite (consisting of at least two elements) or infinite sequence of elements of the set $\mathfrak{F}_{V,A}$, so that the notations of the form $\cap_{i \in I} f_i$ and $\boxplus_{i \in I} f_i$ do not cause any misunderstanding.

4 Analysis of the poset $(\mathfrak{F}_{V,A}, \preceq)$ in terms of lattice theory

The formulas (1) and (2) imply that the poset $(\mathfrak{F}_{V,A}, \preceq)$ is a lower semilattice such that $\inf\{f_1, f_2\} = f_1 \cap f_2$ $(f_1, f_2 \in \mathfrak{F}_{V,A})$. Thus, all basic set-theoretic structures defined on lower semilattices can be transferred to the poset $(\mathfrak{F}_{V,A}, \preceq)$. Let us analyze these structures.

For any non-empty set $S \subseteq \mathfrak{F}_{V,A}$ its lower and upper cones are defined, respectively, using the identities

$$S^{\triangledown} = \{f \in \mathfrak{F}_{V,A} | (\forall f_1 \in S)(f \preceq f_1)\},$$

$$S^{\triangle} = \{f \in \mathfrak{F}_{V,A} | (\forall f_1 \in S)(f \succeq f_1)\}.$$

Lower cones of non-empty subsets of the poset $(\mathfrak{F}_{V,A}, \preceq)$ can be characterized via the following three propositions:

Proposition 5. *For any non-empty subset $S \subseteq \mathfrak{F}_{V,A}$:*

1) the least element of the lower cone $S^{\triangledown}$ is the V-nominative set $0_{V,A}$ with empty domain;

2) the largest element of the lower cone $S^{\triangledown}$ is $\cap_{f \in S} f$.

Proposition 6. *For any non-empty subsets $S_1, S_2 \subseteq \mathfrak{F}_{V,A}$ the following formulas hold:*

(i) $S_1 \subseteq S_2 \Rightarrow S_1^{\triangledown} \supseteq S_2^{\triangledown}$;

(ii) $S_1 \subset S_2 \& \cap_{f_2 \in S_2 \setminus S_1} f_2 \prec \cap_{f_1 \in S_1} f_1 \Rightarrow S_1^{\triangledown} \supset S_2^{\triangledown}$;

(iii) $S_1 \cup S_2 \subseteq \mathfrak{F}_{V,A} \Rightarrow (S_1 \cup S_2)^{\triangledown} = S_1^{\triangledown} \cap S_2^{\triangledown}$.

Proposition 7. *For any $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following formulas hold:*

(i) $\{f_1\}^{\triangledown} \neq \{f_2\}^{\triangledown} \Leftrightarrow f_1 \neq f_2$;

(ii) $\{f_1\}^{\triangledown} \subseteq \{f_2\}^{\triangledown} \Leftrightarrow f_1 \preceq f_2$;

(iii) $\{f_1 \cap f_2\}^{\triangledown} = \{f_1\}^{\triangledown} \cap \{f_2\}^{\triangledown}$;

(iv) $f_2 \not\preceq f_1 \Rightarrow \{f_1 \triangleright f_2\}^{\triangledown} \supseteq \{f_1\}^{\triangledown} \cap \{f_2 \setminus f_1\}^{\triangledown}$;

(v) $f_1|_{\mathrm{Dom} f_1 \cap \mathrm{Dom} f_2} = f_2|_{\mathrm{Dom} f_1 \cap \mathrm{Dom} f_2} \Rightarrow$
$\Rightarrow \{f_1 \triangleright f_2\}^{\triangledown} = \{f_1\}^{\triangledown} \cap \{f_2\}^{\triangledown}$.

Upper cones of non-empty subsets of the poset $(\mathfrak{F}_{V,A}, \preceq)$ can be characterized in the following way:

for any 1-element subset $S = \{f\}$ $(f \in \mathfrak{F}_{V,A})$ the following inequality holds: $S^{\triangle} \neq \emptyset$ (since $f \in \{f\}^{\triangle}$ for any $f \in \mathfrak{F}_{V,A}$).

It is worth to note that $\{0_{V,A}\}^{\triangle} = \mathfrak{F}_{V,A}$. The following proposition is true:

Proposition 8. *For any $(f_1, f_2 \in \mathfrak{F}_{V,A})$ the following formulas hold:*

(i) $\{f_1\}^{\triangle} \neq \{f_2\}^{\triangle} \Leftrightarrow f_1 \neq f_2$;

(ii) $\{f_1\}^{\triangle} \subseteq \{f_2\}^{\triangle} \Leftrightarrow f_1 \preceq f_2$.

The next example illustrates that there exist subsets $S \subseteq \mathfrak{F}_{V,A}$ $(|S| \geq 2)$, such that $S^{\triangle} = \emptyset$.

Example 1. *Let $v \in V$ and $a_1, a_2 \in A$ $(a_1 \neq a_2)$ be fixed elements. We set $S = \{f_1, f_2\}$, where $f_1, f_2 \in \mathfrak{F}_{V,A}$ are V-nominative sets over A such that $Dom f_1 = Dom f_2 = \{v\}$, $f_1(v) = a_1$ and $f_2(v) = a_2$.*

The formula (1) implies that there does not exist any V-nominative set $f \in \mathfrak{F}_{V,A}$, such that $f_1 \preceq f$ and $f_2 \preceq f$. Thus, $S^{\triangle} = \emptyset$.

Now we extract subsets $S \subseteq \mathfrak{F}_{V,A}$ such that $S^{\triangle} \neq \emptyset$.

We will say that elements $f_1, f_2 \in \mathfrak{F}_{V,A}$ are *compatible*, if the identity $f_1|_{\mathrm{Dom} f_1 \cap \mathrm{Dom} f_2} = f_2|_{\mathrm{Dom} f_1 \cap \mathrm{Dom} f_2}$ holds. It is evident that if elements $f_1, f_2 \in \mathfrak{F}_{V,A}$ are compatible, then the following identity holds

$$graph(f_1 \triangleright f_2) = graph(f_1) \cup graph(f_2).$$

Thus we get that for any compatible elements $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following identities hold: $f_1 \triangleright f_2 = f_2 \triangleright f_1 = f_1 \cup f_2$ and $\{f_1 \cup f_2\}^{\triangledown} = \{f_1\}^{\triangledown} \cap \{f_2\}^{\triangledown}$.

A non-empty subset $S \subseteq \mathfrak{F}_{V,A}$ will be called *compatible*, if its elements are pairwise compatible. We denote $\mathfrak{S}^{cmp}_{V,A}$ the set of all compatible subsets of the set $\mathfrak{F}_{V,A}$. The following theorem is true:

Theorem 3. *For any set $\mathfrak{F}_{V,A}$ the following formula holds*

$$(\forall S \subseteq \mathfrak{F}_{V,A})(S \neq \emptyset \Rightarrow (S^{\triangle} \neq \emptyset \Leftrightarrow S \in \mathfrak{S}^{cmp}_{V,A})).$$

Upper cones of elements of the set $\mathfrak{S}^{cmp}_{V,A}$ can be characterized in the following way:

Proposition 9. *For any $S \in \mathfrak{S}^{cmp}_{V,A}$ the following formulas hold:*

(i) $\text{g.l.b.}(S^{\triangle}) = f \Leftrightarrow graph(f) = \cup_{f' \in S} graph(f')$;

(ii) $(\forall S_1, S_2 \subseteq S)(\emptyset \neq S_1 \subseteq S_2 \Rightarrow S_1^{\triangle} \supseteq S_2^{\triangle})$;

(iii) $(\forall S_1, S_2 \subseteq S)(\emptyset \neq S_1 \subset S_2 \&$
$\& \cup_{f_1 \in S_1} graph(f_1) \subset \cup_{f_2 \in S_2 \setminus S_1} graph(f_2) \Rightarrow S_1^{\triangle} \supset S_2^{\triangle})$;

(iv) $(\forall S_1, S_2 \subseteq S)(S_1 \neq \emptyset \& S_2 \neq \emptyset \Rightarrow S_1 \cup S_2 = S_1^{\triangle} \cap S_2^{\triangle})$.

In the poset $(\mathfrak{F}_{V,A}, \preceq)$ any two elements $f_1, f_2 \in \mathfrak{F}_{V,A}$ such that $f_1 \preceq f_2$ define a closed interval $[f_1, f_2] = \{f \in \mathfrak{F}_{V,A} | f_1 \preceq f \preceq f_2\}$. It is evident that $[f_1, f_2] \in \mathfrak{S}^{cnst}_{V,A}$ $(f_1, f_2 \in \mathfrak{F}_{V,A}, f_1 \preceq f_2)$.

The following theorem is true:

Theorem 4. *The algebraic system* $([f_1, f_2], \{\cup, \cap\})$ $(f_1, f_2 \in \mathfrak{F}_{V,A}; f_1 \preceq f_2)$ *is a complete distributive lattice.*

On any closed interval $[f_1, f_2]$ $(f_1, f_2 \in \mathfrak{F}_{V,A}, f_1 \preceq f_2)$ the following unary operation $\mathsf{C}_{[f_1,f_2]}$ can be defined:

$$\mathsf{C}_{[f_1,f_2]}(f) = f' \Leftrightarrow graph(f') = graph(f_2) \backslash graph(f) \cup graph(f_1).$$

The following theorem is true:

Theorem 5. *The algebraic system*

$$([f_1, f_2], \{\cup, \cap, \mathsf{C}_{[f_1,f_2]}\}) \ (f_1, f_2 \in \mathfrak{F}_{V,A}; f_1 \preceq f_2)$$

is a Boolean algebra.

We will say that a mapping $\varphi : \mathfrak{F}_{V,A} \to \mathfrak{F}_{V,A}$ is isotonic on some set $S \subseteq \mathfrak{F}_{V,A}$ $(S \neq \emptyset)$, if the inequality $\varphi(f_1) \preceq \varphi(f_2)$ holds for all $f_1, f_2 \in S$, such that $f_1 \preceq f_2$. It is evident that if $\varphi : \mathfrak{F}_{V,A} \to \mathfrak{F}_{V,A}$ is any mapping isotonic onto some closed interval $[f_1, f_2]$ $(f_1, f_2 \in \mathfrak{F}_{V,A}; f_1 \preceq f_2)$ and the inclusion $\mathrm{Val}\varphi|_{[f_1,f_2]} \subseteq [f_1, f_2]$ holds, then the mapping $\varphi|_{[f_1,f_2]}$ has at least one fixed point.

Let $f_1^{(i)}, f_2^{(i)} \in \mathfrak{F}_{V,A}$ $(i = 1, 2)$ be elements such that $f_1^{(i)} \preceq f_2^{(i)}$. The closed intervals $[f_1^{(1)}, f_2^{(1)}]$ and $[f_1^{(2)}, f_2^{(2)}]$ are *isomorphic*, if there exists a mapping $\varphi : \mathfrak{F}_{V,A} \to \mathfrak{F}_{V,A}$ such that $\varphi|_{[f_1,f_2]}$ is bijection of $[f_1^{(1)}, f_2^{(1)}]$ onto $[f_1^{(2)}, f_2^{(2)}]$ for which the identities

$$\varphi|_{[f_1,f_2]}(f' \cup f'') = \varphi|_{[f_1,f_2]}(f') \cup \varphi|_{[f_1,f_2]}(f'')$$

and

$$\varphi|_{[f_1,f_2]}(f' \cap f'') = \varphi|_{[f_1,f_2]}(f') \cap \varphi|_{[f_1,f_2]}(f'')$$

hold for any $f', f'' \in [f_1^{(1)}, f_2^{(1)}]$.

It is evident that if closed intervals $[f_1^{(1)}, f_2^{(1)}]$ and $[f_1^{(2)}, f_2^{(2)}]$ are isomorphic, then the algebraic systems $([f_1^{(1)}, f_2^{(1)}], \{\cup, \cap\})$ and $([f_1^{(2)}, f_2^{(2)}], \{\cup, \cap\})$, as well as Boolean algebras

$$([f_1^{(1)}, f_2^{(1)}], \{\cup, \cap, \mathsf{C}_{[f_1^{(1)}, f_2^{(1)}]}\}),$$

$$([f_1^{(2)}, f_2^{(2)}], \{\cup, \cap, \mathsf{C}_{[f_1^{(2)}, f_2^{(2)}]}\})$$

are isomorphic.

The following theorem is true:

Theorem 6. *A closed interval* $[f_1, f_2]$ $(f_1, f_2 \in \mathfrak{F}_{V,A}; f_1 \preceq f_2)$ *is isomorphic to the closed interval* $[0_{V,A}, f_2 \backslash f_1]$.

We will say that a closed interval $[0_{V,A}, f]$ is *maximal* in the poset $(\mathfrak{F}_{V,A}, \preceq)$, if $f \in \mathfrak{F}_{V,A}^{(ttl)}$. The following theorem is true:

Theorem 7. *Any two maximal closed intervals in the poset* $(\mathfrak{F}_{V,A}, \preceq)$ *are isomorphic.*

Thus the poset $(\mathfrak{F}_{V,A}, \preceq)$ is a union of the set of overlapping isomorphic maximal closed intervals. At the same time, mappings defining the isomorphism of two intervals differ significantly from each other. Moreover, the structure of the family of these mappings is sufficiently complicated. These circumstances, largely cause high internal complexity of various structures defined on the poset $(\mathfrak{F}_{V,A}, \preceq)$.

5 Conclusions

In the paper a mathematical (algebraic, in essence) formalism intended for investigating the structure of nominative data sets has been proposed. It forms a part of theoretical foundations for unified development of formal methods for automated software design and verification (at least on the level of functional automation testing).

In this context investigation of algebras of programs over nominative data is essential. Investigation of this problem on the basis of the proposed approach may be a topic of the future research.

In the given paper we restricted ourselves to "simple" (flat) nominative data. Elaboration of the proposed approach for hierarchical nominative data is another topic for the future research.

Resolving the above mentioned problems will create a strong algebraic basis for unified development of formal methods intended for

analysis of functions and predicates over nominative data. For investigation of this problem in the future research the approach presented in this paper has been developed.

References

[1] V.N. Redko. *Backgrounds of compositional programming.* Programming, No 3 (1979), pp. 3–13. [in Russian]

[2] M.S. Nikitchenko, S.S. Shkilnjak. *Mathematical logic and algorithms theory.* Kiev National University Press, 2008 [in Ukrainian].

[3] M.S. Nikitchenko, S.S. Shkilnjak. *Applied logic.* Kiev National University Press, 2013 [in Ukrainian].

[4] N.S. Nikitchenko. *A Composition-nominative approach to program semantics.* Technical Report IT-TR 1998-020, Technical Univer sity of Denmark, ISSN 1396-1608, 1998.

[5] A.V. Lamsweerde. *Requirements engineering: from system goals to UML models to software specifications.* Gran Bretaa, Inglaterra, ISBN: 978-470-01270-3, 2009.

[6] J. Woodcock, P.G. Larsen, J. Bicarregui, J. Fitzgerald. *Formal methods: practice and experience.* ACM Computing Surveys, No 4 (2009), pp. 1–36.

[7] V.V. Skobelev. *Problem of checking satisfyability for formulas of solvable theories (a survey).* Proc. of IAMM of NAS of Ukraine, Vol. 26 (2013), pp. 205–221. [in Russian]

[8] W. Ackermann. *Solvable cases of the decision problem.* The Journal of Symbolic Logic, No 1 (1957), pp. 68–72.

[9] D.A. Davey, H.A. Priestly. *Introduction to lattices and order.* Cambridge University Press, 2002.

[10] G. Gratzer. *Lattice theory: foundation.* Springer, Basel AG, 2011.

[11] R. Lidl, G. Pills. *Applied abstract algebra.* Springer, NY, 1998.

Volodymyr G. Skobelev, Ievgen Ivanov,
Mykola Nikitchenko,

Received July 12, 2015

Volodymyr G. Skobelev
V.M. Glushkov Institute of Cybernetics of NAS of Ukraine
40 Glushkova ave., Kyiv, Ukraine, 03187
Phone: +38 063 431 86 05
E–mail: skobelevvg@mail.ru

Ievgen Ivanov
Taras Shevchenko National University of Kyiv
01601, Kyiv, Volodymyrska st, 60
Phone: +38044 2590519
E–mail: ivanov.eugen@gmail.com

Mykola Nikitchenko
Taras Shevchenko National University of Kyiv
01601, Kyiv, Volodymyrska st, 60
Phone: +38044 2590519
E–mail: nikitchenko@unicyb.kiev.ua

Part 3

Natural computing

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Natural Computing: Achievements, Dreams, Limits (Extended Abstract)

(Invited paper)

Gheorghe Păun

Natural computing is sort of a "fashion" in computer science, a long lasting one, motivated, on the one hand, by the limits of what we call "Turing-von Neumann computers" (especially concerning the speed, but also related to other features, such as the ability of learning, energy consumption, the "Turing barrier" as the limit of computability, etc.) and, on the other hand, by the need for new (discrete, information handling, algorithmic) tools for various sciences, especially for biology and biomedicine (but also for many other areas). In this context, important general questions are also raised, such as: what is a computation?, does nature computes?, what means to compute *in a natural way*?, and so on.

There are important achievements of natural computing, at all levels, from mathematical theory to practical applications – but still many problems remain open. (At an elementary technical level, these assertions will be illustrated with facts from DNA and membrane computing.)

New areas of theoretical computer science have been developed, old areas, results, techniques got new incentives, motivations for further research. The applications are numerous and often impressive. Evolutionary computing, with its many branches, continuously diversified, is the most illustrative example. New areas appear somewhat periodically, together with big words and big promises. DNA computing is one good example of this kind.

The main dreams of natural computing are (i) to "break the **NP** barrier" (finding feasible solutions to problems of exponential complexity), (ii) modeling complex objects and phenomena (typical example: modeling the cell, which is considered as the main challenge of bioinformatics after the completion of the Genome Project, (iii) to "break the Turing barrier". The last research area has an established name, *hypercomputing*, many results, a large bibliography, supporters, but also strong critics (such as Martin Davis, who considered hypercomputation a *myth*).

Symmetrical to the hypercomputation concept, the term *fypercomputation* was proposed as a name for the research aiming to find polynomial time solutions to non-polynomial problems (the initial "f" is taken from "fast"). Especially fypercomputability is the goal of natural/unconventional computing, a very important issue from a practical point of view, although it is expected that "computing the uncomputable" could have even more important consequences than proving that $\mathbf{P} = \mathbf{NP}$.

Actually, except in theory, natural computing was not successful in fypercomputation, but in finding approximate/probabilistic solutions to hard problems, or in pushing the feasibility frontiers when looking for exact solutions. Strategies such as trading space for time or making use of a massive parallelism work in principle, but they have drastic practical limits. In turn, evolutionary approaches are the subject of so-called no free lunch theorems.

In short: good motivations, impressive applications, but, still, many dreams not yet reached, big words, new names for old stuff – unfortunately, necessary, in view of the current financing system, but not supported by the reality ("Systems biology: The reincarnation of systems theory applied in biology?", asks O. Wolkenhauer in the title of a paper from *Briefings in Bioinformatics*, 2001).

What is more important (both theoretically and practically) is that there exist several "impossibility theorems", similar to Gödel and Arrow theorems, which limit *in principle* certain performances of natural computing, making impossible the fulfillment of some dreams. The no free lunch theorems were invoked before. Similar results are those of

M. Conrad and R. Gandy, probably similar theorems can be found in the modeling area.

It is possible to have a limit also in what concerns the "meta-dream" of building a mathematical (theory of) biology – some people believe that to this aim essentially new branches of mathematics should be developed, while Auguste Compte, the positivist philosopher, simply believed that this is impossible, useless and harmful for biology...

I am not so pessimistic, but a good degree of lucidity is however necessary...

References

[1] S. Cook. The importance of the P versus NP question. *Journal of the ACM*, 50, 1 (2003), pp. 27–29.

[2] M. Conrad. The price of programmability. In *The Universal Turing Machine: A Half–Century Survey* (R. Herken, ed.), Kammerer and Unverzagt, Hamburg, 1988, pp. 285–307.

[3] B.J. Copeland. Hypercomputation. *Minds and Machines*, 12, 4 (2002), pp. 461–502.

[4] M. Davis. The myth of hypercomputation. In *Alan Turing: The Life and Legacy of a Great Thinker* (C. Teuscher, ed.), Springer, Berlin, 2004, pp. 195-212.

[5] R. Gandy. Church's thesis and principles for mechanisms. *The Kleene Symposium* (J. Barwise et al., eds.), North-Holland, Amsterdam, 1980, pp. 123–148.

[6] Gh. Păun. *Looking for Computers in the Biological Cell. After Twenty Years*. The Publ. House of the Romanian Academy, Bucharest, 2014 (in Romanian).

[7] Gh. Păun, G. Rozenberg, A. Salomaa, eds. *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.

[8] G. Rozenberg, Th. Bäck, J.N. Kok, eds. *Handbook of Natural Computing*. Springer, Berlin, 2012.

[9] D.H. Wolpert, W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1, 67 (1997).

[10] H. Zenil., ed. *A Computable Universe. Understanding and Exploring Nature as Computation*. World Scientific, Singapore, 2013.

[11] The P Systems Website. `http://ppage.psystems.eu`.

Gheorghe Păun Received June 16, 2015

Gheorghe Păun
Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
E–mail: `gpaun@us.es`

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

HPC patterns based implementations of P systems based solutions of hard computational problems

Artiom Alhazov Lyudmila Burtseva Svetlana Cojocaru
Alexandru Colesnicov Ludmila Malahov

Abstract

This is an introductory paper of fresh started research that concerns the development of new HPC parallelization strategies based on intrinsically parallel unconventional computation paradigms. The main goal of this research is to use theses developed parallelization technologies for overcoming modern HPC energy consumption issues. Development of new parallelization technologies will be focused on a particular type of bio-inspired models of computation: the membrane computing paradigm. The solutions for the set of selected problems will be obtained by application of membrane computing. After that these solutions will be decomposed into elementary steps and HPC patterns will be created for each step using ready libraries in order to implement proposed solution on HPC efficiently. A pattern here means an algorithmic skeleton corresponding to a physical hardware architecture as well as the necessary implementation libraries. Basing on the feedback of HPC patterns development, the membrane models would be eventually adjusted in order to obtain simpler and/or more efficient patterns.

1 Introduction

This work concerns the application of intrinsically parallel membrane computing paradigms to express hard problem solutions. The aim of

this application is to use HPC implementations of developed solutions for speeding-up and scalability improving of the existing classical computing approaches.

The increasing heterogeneity and parallelism of emerging parallel computing environment systems such as Grids and Clouds imply that parallelization paradigms must be able to adapt to changes in their requirements. The attempts of traditional approaches solving this modern parallelization problem have obtained so poor results, that the limit of ordinary silicon computing devices was declared. Current research shows, however, that the limit is related to algorithmic support rather than to devices themselves. Overcoming of computation issues described above will be achieved by the development of new parallelization technologies, in particular ones based on intrinsically parallel unconventional computation paradigms.

The underlying idea of current research was proposed and tested on particular problem solution [1]. Although the work [1] proposed just a simulator of particular P system, the authors implemented this simulator as hardware one. Moreover, to use parallel hardware, researchers applied natural parallelization strategies of P system computing. The obtained hardware simulator of the particular P system has significant productiveness of 2×10^7 steps per second.

Our approach is focused on a particular type of models of unconventional computation: the bio-inspired "membrane computing" paradigm, and its models, known as membrane systems or P systems [2], which reproduce the membrane structure of the biological cell. P systems perform calculations using a biologically-inspired process based upon the structure of biological cells. Although inspired by biology, the primary research interest in P systems is concerned with investigating them as a computational model. Membrane computing is a rapidly expanding research area spanning the fields of computer science and engineering, image processing, ecosystems modeling, etc.

P systems offer a new computational modeling framework which integrates the structural and dynamical aspects of cellular systems in a relevant and comprehensive way, while simplifying the formalization necessary to perform mathematical and computational analysis. Mem-

brane computing paradigm is especially attractive because of the versatility and flexibility of the models. P systems have the ability of managing elements from discrete mathematics because the usage of multiset rewriting is close to biological notation. Membrane computing has the algorithmic character that implies easy programmability. P system paradigm presumes natural scalability and massively parallelism, as well as the modularity of the design.

Summarizing above, the application of P systems paradigm for efficient parallelization and computing resources optimization seems very important and attractive. Since P systems are intrinsically highly parallel, P systems based solutions are conceptually different from the traditional ones. The HPC techniques permit to put this parallelism into practice directly. It means HPC implementation of P systems based solutions rather than simulation of the generic model.

The proposed parallelization technologies are presented in form of HPC patterns. In this work we mean *pattern* to be an algorithmic skeleton corresponding to physical hardware architecture as well as the necessary implementation libraries.

Since the mentioned above variety of P system types is so large, the automatic determination of a necessary type for particular problem solution is impossible. Thus the proposed approach does not suppose development of general-purpose patterns. Designing solutions for particular problems, we intend to discover *reusable patterns*. Library of such reusable patterns is supposed to be main practical result of the current research.

The proposed speed-up has sense only for those use cases which are vital to justify costly solutions. The problems of medicine records and imaging, energy or city traffic management, etc., require everyday solving, being unavoidably data-intensive.

This is an introductory paper dedicated to the proposed parallelization technique. More applications and other aspects like estimations of efficiency and consumed resources, synchronization, etc., are subject of further development.

2 HPC patterns bilateral design process

The idea of application of P systems paradigm for efficient parallelization was born at the junction of two domains: HPC and unconventional computing. The issues of HPC implementation by current classical computing urges the researchers to look for new parallelization techniques. From the other side, unconventional computing seeks both the implementation on today's hardware and the solutions of real life problems. So, we suppose that scheme of research will be "bilateral" for these two domains.

As mentioned above, HPC domain researchers are attempting to overcome modern HPC energy consuming issues by the development of new parallelization technologies. In particular, such development concerns HPC programming techniques, which can be assisted by unconventional computing.

Membrane computing, in its turn, is a rapidly expanding research area on crossroads of the fields of computer science and engineering including even practical applications. Although the membrane computers do not exist even as prototype, the results of practical problem solutions obtained by simulators are acceptable. Moreover, an easy programmability of P system paradigm can significantly support new programming techniques design. The parallelism is intrinsic to membrane systems due to multiple reasons of its nature: unlike sequential or some bounded-parallelism models, in P system there is no concept of CPU or global state, but it is rather the rules that are applied "in parallel".

We would like to point out the following guidelines. First, multiple objects evolve in parallel (maximal parallelism is the most studied derivation mode). Second, additional controls, e.g., promoters or inhibitors permit one object to influence an unbounded number of objects. Third, evolution in different regions happens in parallel (distributivity).

Adopting the idea proposed by [1] we intend to develop new parallelization techniques. These techniques will be presented in form of detailed description of P system based reusable HPC patterns obtain-

ing and applying. We set exactly patterns (algorithmic skeletons) as the results because the main idea of P system computing assistance to HPC considers support on programming level. Patterns of parallelism permit developers to skip implementation details and to focus on algorithm structure. The intrinsically parallel algorithm structures of unconventional computing are main diversity that promises the advantage against classic computing. Thus, building a "bridge" between P system computing and HPC consists in process of usage of P system based patterns by HPC application developer. Patterns usage in implementation on HPC means providing application-specific code that customizes the pattern behavior.

The authors of work [1] just tested the idea and did not concern with proposing the methodology. The described research was proceeded in test-and-errors way, but the main aspect is "bilateral" scheme that we adopt and intend to develop further.

The main stages of the research will be as follows.

Firstly, the P system based solution will be developed for the set of selected use cases. Use cases supposed to be selected from the list of problems already known as resolvable by P systems formalism. Another reason of selection is the everyday necessity of present problem solution. Such necessity warrants the usage of expensive HPC computing.

At the second stage the classes of P systems, which are applied for solutions, are analyzed looking for parallelizable features (for example, from the GPU point of view). During the analysis, the behavior and functioning of each P system model will be decomposed into elementary steps. For each obtained elementary step a HPC implementation pattern will be defined. Concentrating on P system based technique, we do intend to research deeply the domain of HPC parallelization pattern development not restricting ourselves by existing techniques. So, at the final step of second stage, basing on the analysis, the existing classes of P systems will be adjusted and, in case of inefficiency, new classes will be defined.

Summarizing the aforesaid, it is evident that the technique of decomposition into elementary steps and the definition of those steps content will be the essence of the proposed research. The initial tech-

nique proposed in research [1] concerns only one particular problem. Solutions of selected problems are supposed to be source of enhancing of techniques of decomposition because they would require different types of P systems.

The results of proposed research will consist generally in methods of generating of P systems based reusable HPC patterns for implementation of particular problem solution. The library of P systems based reusable HPC patterns obtained during research applied to selected problem solutions can be considered as supplementary result.

An ideal previewed results would be generic parallelization algorithms that take P systems as input and produce HPC programs as output.

3 Conclusions

In this paper, the introduction of our new research subject is presented. Presented research concerns the application of intrinsically parallel membrane computing paradigms to express hard problem solutions. The aim of this application is to use HPC implementations of developed solutions for speeding-up and scalability improving of the existing classical computing approaches. It does not mean the creation of unconventional computing simulators or just direct application of existing techniques. Instead, we propose the development of new programming models and paradigms.

The main objective of research proposed in this paper is to obtain a new programming methodology for HPC directly through solving topical, real-life problems.

The fruitfulness of proposed approach is confirmed by existent working example that showed the significant speed-up against classical computation approach.

The efficiency assessment supposes to be expressed as increasing the performance and decreasing the energy consumption.

The main output of the research is previewed in the form of new parallelization strategies based on unconventional computing paradigms.

We concentrated in this paper on presentation of proposed idea. In our following works we will provide more detailed description of technique of decomposition into elementary steps as well as method of presenting and selecting the HPC parallelization patterns.

References

[1] S. Verlan, J. Quiros (2012). *Fast Hardware Implementations of P Systems.* In Membrane Computing - 13th International Conference, CMC 2012, Budapest, Hungary, August 28-31, 2012, Revised Selected Papers.Springer. Lecture Notes in Computer Science, Volume 7762, pp. 404–423.

[2] Gh. Păun. *Membrane Computing. An Introduction.* Springer, 2002.

Artiom Alhazov, Lyudmila Burtseva,
Svetlana Cojocaru, Alexandru Colesnicov,
Ludmila Malahov

Received July 10, 2015

Institute of Mathematics and Computer Science
Str. Academiei 5,
Chişinău, MD-2028,
Moldova
Phone: +373 22 72 59 82
E–mails:
{artiom.alhazov,liudmila.burteva,svetlana.cojocaru,kae,mal}@math.md

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Solving Problem of Graph Isomorphism by Membrane-Quantum Hybrid Model

Artiom Alhazov Lyudmila Burtseva Svetlana Cojocaru
Alexandru Colesnicov Ludmila Malahov

Abstract

This work presents the application of new parallelization methods based on membrane-quantum hybrid computing to graph isomorphism problem solving. Applied membrane-quantum hybrid computational model was developed by authors. Massive parallelism of unconventional computing is used to implement classic brute force algorithm efficiently. This approach does not suppose any restrictions of considered graphs types. The estimated performance of the model is less then quadratic that makes a very good result for the problem of **NP** complexity.

1 Introduction

The present paper concerns application of new computational models based on hybrid of bio-inspired and quantum approaches. In computability theory, a model defines feasible computational operations with their execution time/space. There are many branches of bio-computation: evolution, DNA, swarm, etc. We took as our base the membrane computing formalism, also known as P systems [4].

A P system is a set of mutually inclusive membranes that contain multisets of objects (numbers, strings, or some abstract items) and evolve under some rules. All possible rules are applied in parallel to all possible membranes and objects. This is the membrane parallelism that makes this computation model very powerful.

In our model, membranes can additionally contain quantum machines that perform quantum computations (Fig. 1).

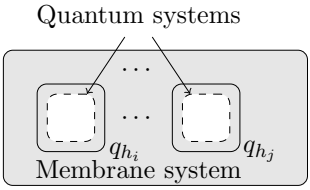


Figure 1. Structure of the hybrid model

The idea of such hybrid computation arises from needs of different domains delivering hard tasks, which are not always satisfactorily solved by existing high performance computational models.

To illustrate the proposed model, we present in this paper a solution of the graph isomorphism problem (GI). Due to its practical applications ranging from chemistry to social sciences, this problem has been solved by many algorithms, both classical and unconventional, still remaining under investigation. Unlike problems which are usually considered as suitable for unconventional computation, GI belongs to **NP**, but is not known to belong to its well-studied subsets like **P** or **NP**-complete. The best classical algorithm complexity is $O(c^{\sqrt{n}\log n})$, where n is the number of vertices in the graph.

Because of this uncertainty of general task, it is divided into several subtasks according graph types (trees, planar graphs, poor-connected, etc.). The majority of mentioned subproblems are proved to be in the integer factorization class. So, existence of polynomial-time quantum algorithm for integer factoring makes GI a good candidate for speedup by a quantum computing [1]. Further developments of quantum computing solutions of GI mostly applied the quantum walk [6]. However, all issues attributable for quantum computation such as "probability" results or exponential growing of system size affect the proposed solutions.

As we said above, we choose the membrane computing formalism. The proposed hybrid model is the usual P system framework supplied by capability to perform quantum computations in its membranes.

Membranes, which are supposed to obtain quantum functionality, just have the specific marks in the P system description. In the marked membrane, the apparition of some specific objects (quantum data, or quantum triggers) starts quantum computation. Specified data become the initial state of the quantum registers. After finishing the quantum computation produces other specific objects (quantum results) in the external membrane.

To provide incorporated quantum functionality in the proposed hybrid model, standard scheme of quantum device [7] proves itself to be sufficient.

Both for membrane and quantum part, we use particular P system formalisms and quantum gates in dependence of the solved problem.

In this paper, the hybrid model solving GI is constructed over *decision P system with active membranes* implementing brute force algorithm. The quantum devices perform only comparison using CNOT, NOT and Toffoli gates.

2 Hybrid Computational Model

2.1 Membrane Subsystem

Membrane systems, or **P systems,** consist of a set of mutually inclusive membranes. The membrane structure μ is a rooted tree, traditionally represented by bracketed expression. For example, $[\ [\ [\ ]_4\ ]_2\ [\ ]_3\ ]_1$ denotes membranes 2 and 3 inside membrane 1, and membrane 4 inside membrane 2. There are many different variants of P systems. Some variants may use static membrane structure, others change it during calculations.

Membranes contain multisets of objects. Objects are numbers, strings, or some abstract items. Different operations over objects may be available. The initial state of a P system is always provided. The initial state is some membrane structure with some multisets inside.

The evolution of a P system is governed by a set of rules. Rules are applicable under certain conditions to change the objects in membranes. All possible rules are applied in parallel to all possible mem-

branes and objects (membrane parallelism). The calculation stops when no rules can be applied.

We will use a decision P system with active membranes. *Decision* means that the alphabet of objects contains symbols `yes` and `no` that represent two possible results of calculation. *P system with active membranes* is defined as a tuple:

$$\Pi = \{O, E, \mu, w_1, \cdots, w_m, e_1, \cdots, e_m, R\}.$$

Here O is the alphabet of objects. $E = \{0, 1, ..., k\}$ is a set of membrane electrical charges, or polarizations. μ is a membrane structure of m membranes labeled by integers; we will denote $H = \{1, \ldots, m\}$ a set of membrane labels. $w_i \in O^*$ and $e_i \in E$, $i \in H$, represent initial content and initial polarization of the i-th membrane. Strings w_i over alphabet O (possibly empty) represent multisets of objects from O. R is a set of rules of the form:

(a) $[a \to v]^i_h$, $a \in O$, $v \in O$, $h \in H$, $i \in E$ (evolution rules, used in parallel in the region of the h-th membrane, provided that the polarization of the membrane is i);

(b) $a[]^i_h \to [b]^j_h$, $a, b \in O$, $h \in H$, $i, j \in E$ (communication rules, sending an object into a membrane and possibly changing the polarization of the membrane);

(c) $[a]^i_h \to []^j_h b$, $a, b \in O$, $h \in H$, $i, j \in E$ (communication rules, sending an object out of a membrane, possibly changing the polarization of the membrane);

(d) $[a]^i_h \to b$, $a, b \in O$, $h \in H$, $i \in E$ (membrane dissolution rules; in reaction with an object, the membrane is dissolved);

(e) $[a]^i_h \to [b]^j_h [c]^k_h$, $a, b, c \in O$, $h \in H$, $i, j, k \in E$ (division rules for elementary membranes not containing other membranes inside; in reaction with an object, the membrane is divided into two membranes with the same label, possibly of different polarizations, and the object specified in the rule is replaced in the two new membranes by possibly new objects).

2.2 Quantum Subsystem

The investigated hybrid model supposes additionally that in any membrane the apparition of some specific objects (quantum data, or quantum triggers) starts a quantum calculation. The said data are available as initial state of the quantum registers. After its termination the quantum calculation produces another specific objects (quantum results) inside the membrane. From the P system point of view, the quantum calculation is a step of the membrane calculation.

Quantum device. We suppose a standard quantum device available for quantum calculations. The quantum device contains qubits organized in quantum registers. It works in three steps: non-quantum (classical) initialization of qubits when they are set in base states; quantum transformation when the qubits are non-observable; non-quantum (classical) measurement that produces the observable result.

Several restriction are imposed over the quantum device. Each qubit contain 0, or 1, or (during quantum calculation) superposition of both. Therefore, the initial data and the result may be regarded as non-negative integers in binary notation. The quantum transformation is linear and reversible. The general rule is that arguments and results are kept in different quantum registers. Another general condition is that the ancillary qubits were not entangled with the argument and the result after the calculation.

The construction of a quantum computer shown in Fig. 2 guarantees this.

3 Interface between Membrane and Quantum Sub-systems

Communications between membrane and quantum sub-systems are performed through input/output signals and triggering (Fig. 3).

We define the hybrid system formally as a tuple

$$\beta = (\Pi, T, T', H_Q, Q_N, Q_M, Inp, Outp, t, q_{h_1}, \cdots, q_{h_m}).$$

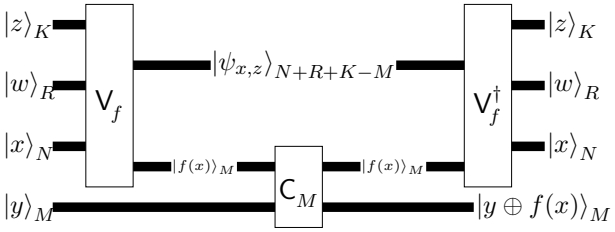


Figure 2. Quantum calculation; initialization and measurement are not shown

Here, Π is a P system, and $H_Q = \{h_1, \cdots, h_m\}$ is a subset of membrane labels in Π used for quantum calculations. T is a trigger and T' is the signal on obtaining the quantum result. Sub-systems $q_{h_1}, \cdots, q_{h_m}$ are the quantum sub-systems associated to the corresponding membranes from H_Q. The rest of the components of the tuple β specify the interaction between Π and q_{h_j}, $1 \leq j \leq m$ (Fig. 3).

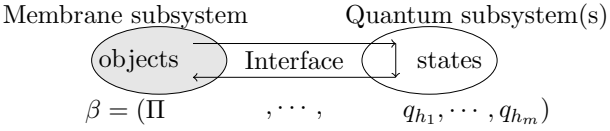


Figure 3. Subsystems and interface in the hybrid model

For simplicity, we assume that the running time of quantum subsystems of the same type is always the same. To keep this time general, we include a timing function $t : H_Q \to \mathbb{N}$: the quantum computation in a sub-system of type q_{h_j} takes $t(h_j)$ membrane steps. It is an open general question how to calculate the timing of quantum calculation with respect to the timing of membrane calculation. We could use as the first rough estimation that quantum calculation takes three steps of membrane calculation (initialization, quantum transformation, measurement).

The input size (in qubits) for quantum systems is given by Q_N :

$H_Q \to \mathbb{N}$. The output size (in bits) for quantum systems is given by $Q_M : H_Q \to \mathbb{N}$.

We would like to define the behavior of β in all possible situations, so we introduce the trigger $T \in O$, where O is the alphabet of Π. The work of a quantum sub-system of type q_{h_j} starts whenever T appears inside the corresponding membrane. Note that we said that q_{h_j} is a type of a quantum sub-system, because in general there may be multiple membranes with label h_j containing quantum sub-systems with the same functionality. The quantum state is initialized by objects from $Inp(h_j) = \{O_{k,h_j,b} \mid 1 \leq k \leq Q_N(h_j),\ b \in \{0,1\}\} \cup \{T\}$, so $Inp : H_Q \to 2^O$ is a function describing the input sub-alphabet for each type of quantum sub-system, the meaning of object $O_{k,h_j,b}$ being to initialize bit k of input by value b. We require that the set of rules satisfies the following condition: any object that may be sent into a membrane labeled h_j must be in $Inp(h_j)$.

The output of quantum sub-systems is returned to the membrane system in the form of objects from $Outp(h_j) = \{R_{k,h_j,b} \mid 1 \leq k \leq Q_M(h_j),\ b \in \{0,1\}\} \cup \{T'\}$, the meaning of object $R_{k,h_j,b}$ being that the output bit k has value b. In case of one-bit output, we often denote it `yes` and `no`.

The result of a quantum sub-system may be produced in the membrane together with object T'.

There are two possibilities to synchronize quantum and membrane levels. We can use a timing function and to wait for the quantum result by organizing the corresponding delay in membrane calculations, or we can wait for appearance of the resulting objects, or the trigger T'. For generality, our model provides both possibilities. The topic needs further investigations.

4 Graphs Isomorphism Problem

GI requires to decide whether two given graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are actually the same graph with relabeling of the vertices.

4.1 Graph Isomorphism: Hybrid Computation

The first graph is represented by objects $a^{(c)}_{i,j,0,0,0}$, where $c = 1$ if the graph has edge (i, j), and $c = 0$ if it does not, $0 \leq i \leq n-1$, $0 \leq j \leq n-1$. The second graph is similarly represented by objects $b^{(c)}_{i,j,0}$. Let $N = \lceil \log_2 n \rceil$ (hence, $n \leq 2^N < 2n$). We construct the following hybrid system.

$$
\begin{array}{rcl}
\beta & = & (\Pi, H_q = \{2\}, n, Inp, Outp = \{\texttt{yes}, \texttt{no}\}, q_2), \text{ where} \\
Inp & = & \{I_{k,b} \mid 0 \leq k \leq 2n^2,\ 0 \leq b \leq 1\}, \\
q_2 & & \text{is a quantum system comparing the first } n^2 \text{ bits with the second} \\
 & & n^2 \text{ bits in } 2N+1 \text{ steps, which is described later, and} \\
\Pi & = & (O, \Sigma, \mu = [\ [\ ]^0_2\ ]^0_1, w_1, w_2, R, 1)
\end{array}
$$

is a decisional P system with active membranes, where

$$
\begin{array}{rcl}
\Sigma & = & \{a^{(c)}_{i,j,0,0,0}, b^{(c)}_{i,j,0} \mid 0 \leq i \leq n-1,\ 0 \leq j \leq n-1, c \in \{0,1\}\}, \\
O & = & \{d_i \mid 1 \leq i \leq nN\} \cup \{p_i \mid 1 \leq u \leq (n+2)N+5\} \cup \{\texttt{yes}, \texttt{no}, X\} \\
 & \cup & \{x_{i,t,k,s} \mid 0 \leq i < n,\ 0 \leq t < n,\ 0 \leq k \leq N,\ 0 \leq s \leq \max(2^{k-1}, 0)\}, \\
 & \cup & \{a^{(c)}_{i,j,t,k,s} \mid -2^N \leq i < n,\ -2^N \leq j < n,\ 0 \leq t \leq n, \\
 & & 0 \leq k \leq N,\ 0 \leq s \leq \max(2^{k-1}, 0),\ c \in \{0,1\}\}, \\
 & \cup & \{b^{(c)}_{i,j,t} \mid 0 \leq i < n,\ 0 \leq j < n,\ 0 \leq t \leq nN+1,\ c \in \{0,1\}\}. \\
w_1 & = & p_1,\ w_2 = d_1 x_{0,0,0,0} \cdots x_{n-1,0,0,0},
\end{array}
$$

and the set R is the union of the following rule groups (together with their explanations): generation, checking, processing the input, and result. Note that the all four groups start working in parallel.

Generation

$$
\begin{array}{ll}
1: & [\ d_i\ ]^e_2 \to [\ d_{i+1}\ ]^0_2 [\ d_{i+1}\ ]^1_2,\ e \in \{0,1\},\ 1 \leq i \leq nN, \\
2: & [\ d_{nN+1}\ ]^e_2 \to [\ ]^0_2 d_{nN+1},\ e \in \{0,1\},
\end{array}
$$

1: creating 2^{nN} membranes and generating for each of them the corresponding nN bits defining the permutations candidates. Other objects may check these bits as membrane polarizations during nN steps (not considering the initial step, where the polarization was 0).

2: After the generation phase, set the polarization to 0.

Checking

3 : $[\ x_{i,t,k,s} \to x_{i,t,k+1,2s+e}\ ]_2^e,\ 0 \le i < n,$
$0 \le t < n,\ 0 \le k \le N-1,\ 0 \le s \le \max(2^{k-1}-1, 0),\ e \in \{0,1\},$

4 : $[\ x_{2s+e,t,N,s} \to \lambda\ ]_2^e,\ 0 \le s \le 2^{N-1}-1,\ 0 \le t < n,\ e \in \{0,1\},$

5 : $[\ x_{i,t,N,s} \to x_{i,t+1,1,0}\ ]_2^e,\ 0 \le i < n,$
$0 \le t < n-1,\ 0 \le s \le 2^{N-1}-1,\ e \in \{0,1\},\ i \ne 2s+e,$

6 : $[\ x_{i,n-1,N,s} \to X\ ]_2^e,\ 0 \le i < n,$
$0 \le s \le 2^{N-1}-1,\ e \in \{0,1\},\ i \ne 2s+e,$

7 : $[\ X\ ]_2^0 \to [\ ]_2^1 X,$

3: Compute the value $\sigma(t)$ of permutation σ for node label t.

4: Erase the label $\sigma(t)$.

5: Continue matching the label.

6: Rename unmatched node labels into X (to make the next step deterministic).

7: Invalid permutation detected. Cancel isomorphism check by setting polarization to 1.

Processing the input

8 : $[\ a^{(c)}_{i,j,t,k,s} \to a^{(c)}_{i,j,t,k+1,2s+e}\ ]_2^e,\ -2^N \le i < n,\ -2^N \le j < n,$
$0 \le t < n,\ 0 \le k \le N-1,\ c \in \{0,1\},\ 0 \le s \le \max(2^{k-1}-1, 0),$

9 : $[\ a^{(c)}_{i,j,t,N,s} \to a^{(c)}_{i',j',t+1,1,0}\ ]_2^e,\ -2^N \le i < n,\ -2^N \le j < n,$
$0 \le t < n,\ c \in \{0,1\},\ 0 \le s \le 2^{N-1}-1,$
$i' = -2s-e-1,\ i = t,\ i' = i$ otherwise,
$j' = -2s-e-1,\ j = t,\ j' = j$ otherwise,

10 : $[\ a^{(c)}_{-i-1,-j-1,n,1,0} \to I_{ni+j,c}\ ]_2^0,\ 0 \le i < n,\ 0 \le j < n,$
$c \in \{0,1\},$

11 : $[\ b^{(c)}_{i,j,t} \to b^{(c)}_{i,j,t+1}\ ]_2^e,\ 0 \le i < n,\ 0 \le j < n,$
$0 \le t \le nN,\ e \in \{0,1\},\ c \in \{0,1\},$

12 : $[\ b^{(c)}_{i,j,nN+1} \to I_{n^2+ni+j,c}\ ]_2^0,\ 0 \le i < n,\ 0 \le j < n,\ c \in \{0,1\},$

8: Compute $\sigma(t)$ for matrix elements.

9: Perform row/column substitution if row/column is t. If so, store the result as a negative index, minus one. In either case, proceed with the next node.

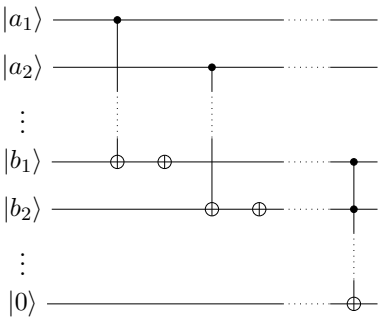


Figure 4. Quantum comparator

11: The input symbols for the second graph wait while the permutations for the first graph are being generated.

10,12: Initialize the quantum subsystem.

Result

13 : $[\ \texttt{yes}\ ]_2^0 \to [\ \ ]_2^1\texttt{yes}$,

14 : $[\ \texttt{yes}\ ]_1^0 \to [\ \ ]_1\texttt{yes}$,

15 : $[\ p_i \to p_{i+1}\ ]_1^0,\ 1 \le i \le (n+2)N+4$,

16 : $[\ p_{(n+2)N+5}\ ]_1^0 \to [\ \ ]_1^1\texttt{no},\ 1 \le i \le (n+2)N+4$.

13: If the quantum subsystem detected a match, send this signal out to the skin.

14: Send the final answer `yes` out, also halting the computation.

15: Wait for the possible answer `yes` to appear.

16: If it did not appear in time, send the final answer `no`.

4.2 Quantum Comparison

Quantum comparison is shown in Fig. 4. It uses CNOT, NOT and Toffoli gates. The result is produced on the qubit initialized by $|0\rangle$ (the lowest in the diagram).

4.3 Notes on Complexity

The classical general algorithm solves GI for graphs of n vertices in time $O(c^{\sqrt{n}\log n})$, were c is a constant [3].

Quantum computation has been widely employed at GI solving during last decade. Initially, users of classic algorithms just applied the Grover method for search between relabeled candidates. But for graphs with n nodes a naive application of Grover search means $O(\sqrt{n!})$ queries, so some other quantum methods have been proposed to improve the efficiency. The most popular of these methods seems to be the quantum walk [6]. The computation complexity of GI solution applying quantum walk is declared for graph with n vertices as $O(n^7)$ for discrete quantum walk [2] and as $O(n^6)$ for continuous one [5].

In the presented GI solution the P system part of computation takes $2\lceil(\log_2 n)\rceil + 1$ steps. Supposing the pure P system computation the algorithm could execute the comparison of each pairs (candidate/pattern) by 2 steps. Totally the pure P system based comparison would take $2n^2$ steps.

We will count the quantum subsystem comparison as 3 steps. So, the whole work time is $(n+2)\lceil(\log_2 n)\rceil + 4$.

5 Conclusions

This paper concerns the application of membrane-quantum hybrid computational model to speed up the classical brute force algorithm solving the problem of graph isomorphism.

Membrane-quantum hybrid computational model is the P system framework with additional quantum functionalities. With this approach, we obtained computation time advancement against both pure membrane and pure quantum solutions, namely: $O(n\log_2 n)$ (hybrid) against $O(n^2)$ (pure membrane) and $O(n^6)$ (pure quantum).

References

[1] S. Dorn. *Quantum Algorithms for Graph and Algebra Problems.* VDM Verlag (2008).

[2] B.L. Douglas, J.B. Wang, *Classical approach to the graph isomorphism problem using quantum walks.* Journal of Physics A: Mathematical and Theoretical 41(7) (2008).

[3] J. Kobler, U. Schoning, J. Toran. *The graph isomorphism problem: its structural complexity.* Birkhauser Verlag (1994).

[4] Gh. Păun. *Membrane Computing. An Introduction.* Springer (2002).

[5] X. Qiang, X. Yang, J. Wu, X. Zhu. *An enhanced classical approach to graph isomorphism using continuous-time quantum walk.* Journal of Physics: A Mathematical and Theoretical 45(4) (2012).

[6] K. Rudinger, J.K. Gamble, E. Bach, M. Friesen, R. Joynt, S.N. Coppersmith. *Comparing algorithms for graph isomorphism using discrete- and continuous-time quantum random walks.* Journal of Computational and Theoretical Nanoscience 10(7), 1653–1661(9) (2013).

[7] C.P. Williams. *Explorations in Quantum Computing.* Springer (2008).

Artiom Alhazov, Lyudmila Burtseva,
Svetlana Cojocaru, Alexandru Colesnicov,
Ludmila Malahov

Received June 17, 2015

Institute of Mathematics and Computer Science
Str. Academiei 5,
Chişinău, MD-2028,
Moldova
Phone: +373 22 72 59 82
E–mail: {artiom.alhazov,lyudmila.burtseva,svetlana.cojocaru,kae,mal}@math.md

Part 4

Theoretical issues in automated reasoning

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Admissibility, compatibility, and deducibility in first-order sequent logics

Alexander Lyaletski

Abstract

The paper is devoted to the notions of admissibility and compatibility and their influence on deducibility in different sequent logics including first-order classical and intuitionistic ones as well as their modal extensions. Results on the coextensivity of the proposed sequent calculi and usual Gentzen and Kanger sequent calculi are given.

Keywords: First-order classical logic, first-order intuitionistic logic, first-order modal logic, sequent calculus, deducibility, admissibility, compatibility, coextensivity, validity.

1 Introduction

Investigations in computer-oriented reasoning gave rise to the appearance of various machine methods for the proof search in first-order logics. Particularly, Gentzen sequent calculi [1] modified for their implementation on a computer have found many applications in computer science. But in the case of classical logic their practical application as a logical technique (without preliminary skolemization) of the intelligent systems has not received wide use: preference is usually given to the resolution-type methods. This is explained by higher efficiency of the resolution-type methods as compared to sequent calculi, which is mainly connected with different possible orders of the quantifier rule applications in sequent calculi while resolution-type methods, due to skolemization, are free from this deficiency.

In its turn, the deduction process in sequent calculi reflects sufficiently well natural theorem-proving methods which, as a rule, do

not include preliminary skolemization so that reasonings are performed within the scope of the signature of the initial theory. This feature of sequent calculi becomes important when some interactive mode of proof is developed since it is preferable to present the output information concerning the proof search in the form usual for man. Besides, preliminary skolemization is not a valid operation for many non-classical logics including intuitionistic one while many of such logics have a wide application in solving reasoning problems. That is the problem of the efficient quantifier manipulation makes its appearance.

When quantifier rules are applied, some substitution of selected terms for variables is made. To do this step of deduction sound, certain restrictions are put on the substitution. A substitution, satisfying these restrictions, is said to be admissible. Here we investigate the classical notion of an admissible substitution and show how it can be modified so that efficient sequent calculi can be finally obtained both for classical and non-classical logics using the so-called compatibility (if necessary). (For simplicity, we restrict us by a detailed enough consideration of classical and intuitionistic logics without equality briefly discussing a possibility to extend them to the equality and modal cases.)

For our purpose, we use modifications of the calculi LK and LJ without equality from [1] and denote them by mLK and mLJ respectively. Moreover, we extend mLK and mLJ in a certain way for logics with equality and modal rules. At that, we don't touch upon any procedure of selection of propositional rules and terms substituted, focusing our attention on quantifier handling only. Also note that in contrary to [1] and [2], the antecedents and succedents of all the sequents under consideration are assumed to be multisets. As usual, inference search in any calculus is of the form of a so-called inference tree "growing" from bottom to top in accordance with counter-applying inference rules. An inference tree all leaves of which are axioms is called a proof tree.

2 Genzen's notion of admissibility

Classical quantifier rules, substituting arbitrary structure terms when applied from bottom to top, are usually of the following form slightly

distinguished from the one given in [1]:

$$\frac{\Gamma, A[t/x] \to \Delta}{\Gamma, \forall x A \to \Delta}\ (\forall : left) \qquad \frac{\Gamma \to A[t/x], \Delta}{\Gamma \to \exists x A, \Delta}\ (\exists : right)$$

where the term t is required to be free for the variable x in the formula A and $A[t/x]$ is the result of the simultaneous replacement in A of x by t. This restriction of the substitution of t for x gives Gentzen's (classical) notion of an admissible substitution, which proves to be sufficient for the needs of the proof theory. But it becomes useless from the point of view of efficiency of computer-oriented theorem-proving methods. It is clear from the following example.

Consider a sequent $A_1, A_2 \to B$, where A_1 is $\forall x_1 \exists y_1 (R_1(x_1) \vee R_2(y_1))$, A_2 is $\forall x_2 \exists y_2 (R_1(y_2) \supset R_3(x_2))$, and B is $\exists x_3 \forall y_3 (R_2(x_3) \vee R_3(y_3))$. The provability of this sequent in calculus LK will be established below, while here we notice that quantifier rules must be applied to all the quantifiers occurring in A_1, A_2, and B. Therefore, classical notion of admissible substitution yields 90 (= 6!/(2!*2!*2!)) different orders of the quantifier rule applications to $A_1, A_2 \to B$. It is clear that resolution type methods allow avoiding this redundant work.

3 Kanger's notion of admissibility

To optimize procedure of the applications of quantifier rules, S.Kanger suggested in [3] his Gentzen-type calculus, denoted here by K. In calculus K a "pattern" of a deduction tree is first constructed with the help of special variables, the so called parameters and dummies. At some times an attempt is made to convert a "pattern" into a proof tree to complete the deduction process. In case of failure, the process is continued. The main difference between K and LK consists in a special modification of the above quantifier rules and in a certain splitting (in K) of the process of the "pattern" construction into stages. In K the rules $(\forall : left)$ and $(\exists : right)$ are of the following form:

$$\frac{\Gamma, A[d/x] \to \Delta}{\Gamma, \forall x A \to \Delta}\ d/t_1, ..., t_n \qquad \frac{\Gamma \to A[d/x]\Delta}{\Gamma \to \exists x A, \Delta}\ d/t_1, ..., t_n$$

where $t_1, \ldots, t_n$ are terms occurring in the conclusion of the rules, d is a dummy, and $d/t_1, \ldots, t_n$ denotes that when an attempt is made to convert "pattern" into a proof tree, the dummy d must be replaced by one of the terms $t_1, \ldots, t_n$. The replacement of dummies by terms is made in the end of every stage, and at every stage the rules are applied in a certain order. This scheme of the deduction construction in calculus K leads to a notion of the Kanger-admissible substitution, which is more efficient than the classical one. Thus in the above example it yields only 6 (=3!) variants of different possible orders of the quantifier rule applications (but none of these variants is preferable). Despite this, the Kanger-admissible substitutions still did not allow attaining the efficiency comparable with that when the skolemization is made. It is due to the fact that, as in case of the classical admissible substitution, it is required to select a certain order of the quantifier rule applications when an input sequent is deduced, and, if it proves to be unsuccessful, the other order of applications is tried, and so on.

4 New notion of admissibility

For constructing the modification mLK of calculus LK from [1], let us introduce a new notion of an admissible substitution in order to get rid of the dependence of the deduction efficiency in sequent calculi on different possible orders of quantifier rule applications. The main idea is to determine, proceeding from quantifier structures of formulas of an input sequent and a substitution under consideration, would there exists a sequence of desired quantifier rules applications. (This notion was used in [4] in slightly modified form for another purpose.)

We assume that besides usual variables there are two countable sets of special variables, namely of parameters and dummies.

A *substitution* is defined as a finite (maybe, empty) set of ordered pairs, every of which consists of a variable and a term and is written in the form t/x, where x is a *variable* and t a *term of substitution* [5]. For a sequent tree D, $D \cdot s$ denotes the result of the simultaneous replacement of all the variables of s by the corresponding terms of s.

Let P be a set of sequences of parameters and dummies and s a

substitution. Put $T(P,s) = \{\langle z,t,p\rangle : z$ is a variable of s, t a term of s, $p \in P$, and z lies in p to the left of some parameter from $t\}$. The substitution s is said to be *admissible* for P if and only if (1) the variables of s are only dummies and (2) in $T(P,s)$ there are no elements $\langle z_1,t_1,p_1\rangle,\ldots,\langle z_n,t_n,p_n\rangle$ such that $t_2/z_1 \in s,\ldots,t_n/z_{n-1} \in s, t_1/z_n \in s$ $(n>0)$.

5 Admissibility and classical deducibility

As in the case of calculus LK, its modification mLK deals with formulas, except that in mLK every formula from a sequent has a (possibly, empty) sequence of parameters and dummies. Thus, it is convenient to define calculus mLK with help of the pairs $\langle p, A\rangle$, where A is a formula and p a sequence (word) of parameters and dummies. Also, it will be assumed that the empty sequence is always added to all formulas from an initial sequent (that is, from a sequent to be proved).

The rules of the calculus mLK are the following:

Propositional rules:

$$\frac{\Gamma, \langle p,A\rangle, \langle p,B\rangle \to \Delta}{\Gamma, \langle p, A\wedge B\rangle \to \Delta} \quad \frac{\Gamma \to \langle p,A\rangle, \Delta \quad \Gamma \to \langle p,B\rangle, \Delta}{\Gamma \to \langle p, A\wedge B\rangle, \Delta}$$

$$\frac{\Gamma, \langle p,A\rangle \to \Delta \quad \Gamma, \langle p,B\rangle \to \Delta}{\Gamma, \langle p, A\vee B\rangle \to \Delta} \quad \frac{\Gamma \to \langle p,A\rangle, \Delta}{\Gamma \to \langle p, A\vee B\rangle, \Delta} \quad \frac{\Gamma \to \langle p,B\rangle, \Delta}{\Gamma \to \langle p, A\vee B\rangle, \Delta}$$

$$\frac{\Gamma, \langle p,A\rangle \to \langle p,B\rangle, \Delta \quad \Gamma, \langle p,B\rangle \to \Delta}{\Gamma, \langle p, A\supset B\rangle \to \Delta} \quad \frac{\Gamma, \langle p,A\rangle \to \langle p,B\rangle, \Delta}{\Gamma \to \langle p, A\supset B\rangle, \Delta}$$

$$\frac{\Gamma, \langle p,A\rangle \to \Delta \quad \Gamma, \langle p,B\rangle \to \Delta}{\Gamma, \langle p, A\vee B\rangle \to \Delta} \quad \frac{\Gamma \to \langle p,A\rangle, \Delta}{\Gamma \to \langle p, A\vee B\rangle, \Delta} \quad \frac{\Gamma \to \langle p,B\rangle, \Delta}{\Gamma \to \langle p, A\vee B\rangle, \Delta}$$

$$\frac{\Gamma \to \langle p,A\rangle, \Delta}{\Gamma, \langle p, \neg A\rangle \to \Delta} \quad \frac{\Gamma, \langle p,A\rangle \to \Delta}{\Gamma \to \langle p, \neg A\rangle, \Delta}$$

Contraction rules:

$$\frac{\Gamma, \langle p,A\rangle, \langle p,A\rangle \to \Delta}{\Gamma, \langle p,A\rangle \to \Delta}\ (Con\to) \quad \frac{\Gamma \to \langle p,A\rangle, \langle p,A\rangle \Delta}{\Gamma \to \langle p,A\rangle, \Delta}\ (\to Con)$$

Quantifier rules:

$$\frac{\Gamma, \langle pd, A[d/x]\rangle \to \Delta}{\Gamma, \langle p, \forall x A\rangle \to \Delta}\ (\forall : left') \qquad \frac{\Gamma \to \langle pz, A[z/x]\rangle, \Delta}{\Gamma \to \langle p, \forall x A\rangle, \Delta}\ (\forall : right')$$

$$\frac{\Gamma, \langle pz, A[z/x]\rangle \to \Delta}{\Gamma, \langle p, \exists x A\rangle \to \Delta}\ (\exists : left') \qquad \frac{\Gamma \to \langle pd, A[d/x]\rangle, \Delta}{\Gamma \to \langle p, \exists x A\rangle, \Delta}\ (\exists : right')$$

Here d is a new dummy, z is a new parameter, p is a sequence of parameters and dummies, Γ and Δ are arbitrary multisets of pairs, consisting of sequences (of dummies and parameters) and formulas, A and B are arbitrary formulas.

In what follows, the establishing of the deducibilty of a sequent $A_1,\ldots,A_m \to B_1,\ldots,B_n$ in LK is replaced by the establishing of the deducibilty of the so-called *initial sequent* $\langle, A_1\rangle,\ldots,\langle, A_m\rangle \to \langle, B_1\rangle,\ldots,\langle, B_n\rangle$, but yet in mLK or its modifications ($A_1,\ldots,A_m$, $B_1,\ldots,B_n$ are formulas of the first-order language).

Applying first a rule from bottom to top to a sequent under consideration and afterwards to its "heirs", and so on, we finally obtain a so-called *inference tree* for this sequent.

Let D be an inference tree in mLK and s a substitution. If all the leaves of $D \cdot s$ are axioms, then D is called a *latent proof tree* in mLK w.r.t. s.

The main result concerning the calculus mLK is as follows.

Theorem 1. *Let $A_1,\ldots,A_m, B_1,\ldots,B_n$ be formulas of the first-order language. The sequent $A_1,\ldots,A_m \to B_1,\ldots,B_n$ is deducible in LK if and only if there exist an inference tree D in mLK for the initial sequent $\langle, A_1\rangle, \ldots, \langle, A_m\rangle \to \langle, B_1\rangle,\ldots,\langle, B_n\rangle$ and a substitution s of terms without dummies for all the dummies of D such that: (1) D is a latent proof tree in mLK w.r.t. s and (2) s is an admissible substitution for the set of all the sequences of parameters and dummies from D.*

Proof. (=>) Let D be a proof tree for the input sequent $\langle, A_1\rangle, \ldots, \langle, A_m\rangle \to \langle, B_1\rangle,\ldots,\langle, B_n\rangle$ in the calculus mLK, and s be a substitution, which converts all the leaves of D into axioms and is admissible for the set P of all sequences of parameters and dummies from D. Without any loss of generality, we may assume that terms of s do not contain

dummies for otherwise these dummies could be replaced by a special constant, say, c_0. Since s is admissible for P, it is possible to construct the following sequence p consisting of parameters and dummies which form the sequences of P:

(i) every $p' \in P$ is a subsequence of p and

(ii) s is admissible for $\{p\}$ (i.e. there is no an element $\langle z, t, p\rangle \in T(p, s)$ such that $t/z \in s$.

Such a sequence p may be generated, for example, by the convolution algorithm from [4], applied to a list of all the sequences from P provided that in the convolution algorithm parameters are treated as existence quantifiers, and dummies as universal quantifiers. The property (i) of the sequence p and definition of the propositional and quantifier rules permit to make the following assumption:

When D was constructed, propositional, contraction, and quantifier rules were applied (from bottom to top) in the order that corresponds to looking through p from the left to right: i.e. when the first quantifier rule was applied, the first variable (a parameter or a dummy) of p was generated, when the second quantifier rule was applied, the second variable of p was generated, and so on.

Now it is possible to convert the tree D into a proof tree D' for the input sequent $A_1, \ldots, A_m \to B_1, \ldots, B_n$ in calculus LK. To do this, let us "repeat" the process of the construction of D in the above-given order p and execute the following transformations:

1) Suppose that in a processed node of D one of the following rules was applied:

$$\frac{\Gamma, \langle pd, A[d/x]\rangle \to \Delta}{\Gamma, \langle p, \forall x A\rangle \to \Delta}\ (\forall : left') \text{ or } \frac{\Gamma \to \langle pd, A[d/x]\rangle, \Delta}{\Gamma \to \langle p, \exists x A\rangle, \Delta}\ (\exists : right')$$

and $t/d \in s$ for some term t. The term t is free for d in A, because the order of applications of quantifier rules is reflected by p, and the property (ii) is satisfied. Hence, the admissibility in the classical (Gentzen) sense will be observed when the above rules $(\forall : left')$ and $(\exists : right')$ are replaced in D by the rules $(\forall : left)$ and $(\exists : right)$ of LK:

$$\frac{\Gamma, A[t/x] \to \Delta}{\Gamma, \forall x A \to \Delta}\ (\forall : left) \text{ or } \frac{\Gamma \to A[t/x]\Delta}{\Gamma \to \exists x A, \Delta}\ (\exists : right)$$

and all occurrences of d in D are replaced by t.

2) In other cases the rules of the calculus mLK are replaced in D by their analogues from LK by a simple deleting of sequences of parameters and dummies from these rules. It is evident that D' is a deduction tree in the calculus LK. Furthermore, the way of conversion of D into D' allows making the conclusion that leaves of D' are axioms of the calculus LK. Thus, D' is a proof tree for the initial sequent A_1, ..., $A_m \to B_1, \ldots, B_n$.

(<=) Let D' be a proof tree for the initial sequent A_1, ..., $A_m \to B_1, \ldots, B_n$ in LK. Convert D' into a tree D, which, as it can be seen below, is a proof tree for the initial sequent $\langle, A_1\rangle$, ..., $\langle, A_m\rangle \to \langle, B_1\rangle, \ldots, \langle, B_n\rangle$ in mLK. For this purpose "repeat" (from bottom to top) a process of construction of D', replacing in D' every rule application by its analogue in mLK and subsequently generating a substitution s. (Initially s is the empty substitution.)

1) If an applied rule is one of the following:

$$\frac{\Gamma, A[t/x] \to \Delta}{\Gamma, \forall x A \to \Delta}\ (\forall : left) \quad \text{or} \quad \frac{\Gamma \to A[t/x]\Delta}{\Gamma \to \exists x A, \Delta}\ (\exists : right)$$

then it is replaced by

$$\frac{\Gamma, \langle pd, A[d/x]\rangle \to \Delta}{\Gamma, \langle p, \forall x A\rangle \to \Delta}\ (\forall : left') \text{ or } \frac{\Gamma \to \langle pd, A[d/x]\rangle, \Delta}{\Gamma \to \langle p, \exists x A\rangle, \Delta}\ (\exists : right')$$

accordingly with adding t/d to the existing substitution s, where d is a new dummy, and with substituting d for those occurrences of t into "heirs" of the formula $A[t/x]$, which appeared as a result of applying of a replaced rule inserting the term t.

2) In all other cases replacement of the rules of LK by the rules of mLK is evident. (Note that $\langle, A_1\rangle$, ..., $\langle, A_m\rangle \to \langle, B_1\rangle, \ldots, \langle, B_n\rangle$ is declared as input sequent of D.) The rules $(\forall : left)$ and $(\exists : right)$ may be considered as those inserting new parameters). Since D' is a proof tree in the calculus utilizing the classical notion of an admissible substitution, then it is clear that the finally generated substitution s is admissible (in the new sense) for a set of all sequences of parameters

and dummies from D. Therefore, D is a proof tree for the initial sequent $\langle, A_1\rangle, \ldots, \langle, A_m\rangle \to \langle, B_1\rangle, \ldots, \langle, B_n\rangle$ in mLK. □

To demonstrate the deduction technique, consider the sequent $A_1, A_2 \to B$ from the above-given example and establish its deducibility in the calculus LK. To do this, construct a proof tree for the initial sequent $\langle, A_1\rangle,\langle, A_2\rangle \to \langle, B\rangle$ in mLK and use Theorem 1.

Applying to the initial sequent first the rule $(\to Con)$ and then only quantifier rules in any order we can deduce the following sequent: $\langle d_1 z_1, R_1(x_1) \vee R_2(y_1)\rangle$, $\langle d_2 z_2, R_1(y_2) \supset R_3(x_2)\rangle \to \langle d_3 z_3, R_2(d_3) \vee R_3(x_3)\rangle$, $\langle d_4 z_4, R_2(d_4) \vee R_3(x_4)\rangle$, where $d_1, \ldots, d_4$ are dummies, $z_1, \ldots, z_4$ parameters.

Now let us apply propositional rules to the last sequent as long as they are applicable. As a result, we get the inference tree, D. If we generate the substitution $s = \{z_2/d_1, z_3/d_2, c_0/d_3, z_1/d_4\}$ (c_0 is a special constant), then we can draw the following conclusions concerning s and D: (1) every leaf from D is transformed into an axiom by applying of s to it and (2) s is admissible for the set of all sequences of dummies and parameters from D.

So, in accordance with Theorem 1 the sequent $A_1, A_2 \to B$ is deducible in the calculus LK.

Draw your attention to the fact that the selection of an order of the quantifier rules applications in mLK is immaterial; it can be any.

6 Compatibility and intuitionistic deducibility

The intuitionistic calculus LJ is distinguished from LK by that the succedent of any sequent deduced in LJ should contain no more than one formula [1]. In this connection it seems that this restriction putting on mLK can lead to a correct intuitionistic modification of the classical calculus mLK, say, mLJ. Unfortunately, it is not so, and the following example demonstrates this fact.

Consider the sequent $\neg\forall P(x) \to \exists y \neg P(y)$. Obviously, it is deducible in LK while it *is not deducible* in LJ.

Let us construct a proof tree D in mLK for it:

$$\dfrac{\dfrac{\dfrac{\dfrac{\langle d, P(d)\rangle \to \langle z, P(z)\rangle}{\langle d, P(d)\rangle \to \langle, \forall x P(x)\rangle}}{\langle, \neg\forall x P(x)\rangle, \langle d, P(d)\rangle \to}}{\langle, \neg\forall x P(x)\rangle \to \langle d, \neg P(d)\rangle}}{\langle, \neg\forall x P(x)\rangle \to \langle, \exists y \neg P(y)\rangle}$$

where d is a dummy and z a parameter.

Denote it by D and consider the substitution $s = \{z/d\}$. It converts the first sequent of D into an axiom and is admissible for D. By Theorem 1, the sequent $\neg\forall P(x) \to \exists y \neg P(y)$ is deducible in LK.

The succedent of any sequent in D contains only one formula. Therefore, the usage of only the notion of admissibility is not enough for providing the "sound" deducibility in mLJ for the sequents deducible in LJ and only in it.

This situation can be corrected with the help of the notion of the so-called *compatibility* of a constructed proof tree with a selected substitution [6]. Because of the paper size limit, this notion will not be detailed below. We note simply that after introducing both the notions of admissibility and compatibility in mLJ, they correlate with each other in such a way that provide the soundness (and completeness) of inference search. For example, the above-given tree D for the sequent $\neg\forall P(x) \to \exists y \neg P(y)$ *is not compatible* with the unique "reasonable" substitution $s = \{z/d\}$, which implies that $\neg\forall P(x) \to \exists y \neg P(y)$ is not deducible in LJ.

The following result takes place for intuitionistic logic.

Theorem 2. *Let $A_1, \ldots, A_m, B_1, \ldots, B_n$ be formulas of the first-order language. The sequent $A_1, \ldots, A_m \to B_1, \ldots, B_n$ is deducible in LJ if and only if there exist an inference tree D in mLJ for the initial sequent $\langle, A_1\rangle, \ldots, \langle, A_m\rangle \to \langle, B_1\rangle, \ldots, \langle, B_n\rangle$ and a substitution s of terms without dummies for all the dummies of D such that: (1) D is a latent proof tree in mLJ w.r.t. s, (2) s is an admissible substitution for the set of all sequences of parameters and dummies from D, and (3) D is compatible with s.*

Draw your attention to the fact that Theorems 1 and 2 are distinguished by only the existence of (3) in the wording of Theorem 2.

7 Admissibility, compatibility, deducibility in equality and modal extensions

Let $\mathrm{LK}^{\approx}$ and $\mathrm{LJ}^{\approx}$ be, respectively, the calculi LK and LJ extended to the case of classical and intuitionistic logics with equality (denoted by $\approx$) by means of introducing in them the Kanger equality rules from [3].

Let us introduce in mLK and mLJ the following modifications of the Kanger equality rules (denoting these equality extensions by $\mathrm{mLK}^{\approx}$ and $\mathrm{mLJ}^{\approx}$ respectively):

$$\frac{\Gamma|^{t'}_{t''},\ \langle p, t' \approx t''\rangle \to \Delta|^{t'}_{t''}}{\Gamma,\ \langle p, t' \approx t''\rangle \to \Delta} \qquad \frac{\Gamma|^{t'}_{t''},\ \langle p, t'' \approx t'\rangle \to \Delta|^{t'}_{t''}}{\Gamma,\ \langle p, t'' \approx t'\rangle \to \Delta}$$

where the terms t' and t'' do not contain dummies and $\Gamma|^{t'}_{t''}$ and $\Delta|^{t'}_{t''}$ are the results of the simultaneous replacement of t' by t'' in Γ and Δ respectively.

As in [3], in $\mathrm{mLK}^{\approx}$ and $\mathrm{mLJ}^{\approx}$ the defined equality rules are applied in inference search last of all, i.e. when it seems impossible to construct such a tree D without applying equality rules and select such a substitution s that the conditions (1), (2), and (3) from Theorems 1 and 2 are satisfied.

Let D be an inference tree constructed in $\mathrm{mLK}^{\approx}$ ($\mathrm{mLJ}^{\approx}$) without applying equality rules and s a substitution. Suppose that after subsequent applying of only the equality rules to all the leaves of $D \cdot s$ not being axioms, then to their "heirs", and so on, an inference tree containing only axioms is produced. Then D is called a *latent proof tree* in $\mathrm{mLK}^{\approx}$ ($\mathrm{mLJ}^{\approx}$) w.r.t. s.

Theorem 3. *Let $A_1, \ldots, A_m, B_1, \ldots, B_n$ be formulas of the first-order language. The sequent $A_1, \ldots, A_m \to B_1, \ldots, B_n$ is deducible in $LK^{\approx}$ ($LJ^{\approx}$) if and only if there exist an inference tree D in $mLK^{\approx}$ ($mLJ^{\approx}$) for the initial sequent $\langle, A_1\rangle, \ \ldots, \ \langle, A_m\rangle \to \langle, B_1\rangle, \ldots, \langle, B_n\rangle$ and a substitution s of terms without dummies for all the dummies of D such that: (1) D is a latent proof tree in $mLK^{\approx}$ ($mLJ^{\approx}$) w.r.t. s, (2) s is an admissible substitution for the set of all the sequences of parameters and dummies from D, and, in the case of $mLJ^{\approx}$, (3) the tree D is compatible with s.*

Our way of the constriction of modal calculi has a certain correlation with the papers [7] as well as [8], where necessary modal rules are simply added to Gentsen's calculi LK and LJ.

Doing the same for LK and LJ and LK$^{\approx}$ and LJ$^{\approx}$, we define their modal extensions LK+Mod$_m$, LJ+Mod$_m$, LK$^{\approx}$+Mod$_m$, and LJ$^{\approx}$+Mod$_m$, where Mod$_m$ is a set of modal rules.

As to modal rules adding to LK, LJ, LK$^{\approx}$, and LJ$^{\approx}$ if necessary, any such a modal rule is considered to be of the following form:

$$\frac{\Gamma, \Phi_1, \ldots, \Phi_k \to \Psi_1, \ldots, \Psi_r, \Delta}{\Gamma, \bigcirc_1(\Phi_1), \ldots, \bigcirc_k(\Phi_k) \to \bigcirc'_1(\Psi_1), \ldots, \bigcirc'_r(\Psi_r), \Delta}$$

where $\bigcirc_1, \ldots, \bigcirc_r, \bigcirc'_1, \ldots, \bigcirc'_r$ are modal operators and Φ_1, ..., Φ_k, Ψ_1, ..., Ψ_r multisets of formulas (containing, possibly, modal operators). For example, it makes possible to determine the calculus GK or GS4 from [8] containing the standard modal operators $\Box$ and $\Diamond$.

Any modal rule of this form transformed into the corresponding modal rule of the following form (that can be introduced in any of the calculi mLK, mLJ, mLK$^{\approx}$, and mLJ$^{\approx}$):

$$\frac{\Gamma', \langle p_1, \Phi_1\rangle, ..., \langle p_k, \Phi_k\rangle \to \langle q_1, \Psi_1\rangle, ..., \langle q_r, \Psi_r\rangle, \Delta'}{\Gamma', \langle p_1, \bigcirc_1(\Phi_1)\rangle, ..., \langle p_k, \bigcirc_k(\Phi_k)\rangle \to \langle q_1, \bigcirc'_1(\Psi_1)\rangle, ..., \langle q_r, \bigcirc'_r(\Psi_r)\rangle, \Delta'}$$

where, $p_1, \ldots, p_k, q_1, \ldots, q_r$ are sequences of dummies and parameters.

Draw your attention to that any such rule satisfies the subformula property, which makes obvious the following proposition in virtue of Theorems 1, 2, and 3.

Theorem 4. *Let $A_1, \ldots, A_m, B_1, \ldots, B_n$ be formulas of the first-order language containing, possibly, modal operators. The sequent $A_1, \ldots, A_m \to B_1, \ldots, B_n$ is deducible in LK+Mod$_m$ (LJ+Mod$_m$, LK$^{\approx}$+Mod$_m$, LJ$^{\approx}$+Mod$_m$) if and only if there exist an inference tree D in LK+Mod$_m$ (LJ+Mod$_m$, LK$^{\approx}$+Mod$_m$, LJ$^{\approx}$+Mod$_m$) for the initial sequent $\langle, A_1\rangle, \ldots, \langle, A_m\rangle \to \langle, B_1\rangle, \ldots, \langle, B_n\rangle$ and a substitution s of terms without dummies for all the dummies of D such that: (1) D is a latent proof tree in LK+Mod$_m$ (LJ+Mod$_m$, LK$^{\approx}$+Mod$_m$, LJ$^{\approx}$+Mod$_m$) w.r.t. s, (2) s is an admissible substitution for the set of all the sequences of parameters and dummies from D, and, in the cases of LJ+Mod$_m$ and LJ$^{\approx}$+Mod$_m$, (3) the tree D is compatible with s.*

The Kanger calculus K without equality is coextensive with the Gentzen calculus LK. It is easy to see that all the above-described constructions made for LK can be transferred to the case of K producing an analogue of mLK for K and its intuitionistic modification as well as their equality and modal extensions retaining the results on coextensivity.

Taking into consideration all the above-given theorems, we can obtain the soundness and completeness theorem for any of our calculi if and only if this theorem takes place for its Gentzen or Kanger analogue. For example, we conclude that the validity of a formula F in classical (intuitionistic) logic with equality is equivalent to the deductibility of the initial sequent $\to \langle , F\rangle$ in $\mathrm{mLK}^{\approx}$ ($\mathrm{mLJ}^{\approx}$).

8 Conclusion

The research presented in this paper demonstrates that the usage of the introduced notions of admissibility and compatibility gives a good enough decision of the problem of quantifier handling in first-order logics. They are easily built-in into the Gentzen calculi LK and LJ, which shows that there is a good basis for constructing computer-oriented sequent calculi for classical and intuitionisticl logics as well as for their equality and modal extensions. At that, the questions of the machine implementation of such sequent calculi are not considered because the construction of efficient calculi requires optimizing the order of the propositional rule applications and selecting a method for generating a substitution which can produce a latent proof tree. Bypassing details observe that the Robinson unification algorithm combined with the new notion of admissibility is suitable for generating such substitutions.

This approach to the construction of methods for inference search in first-order logics corresponds well to a modern vision of the so-called Evidence Algorithm, EA, advances by V. M. Glushkov as early as 1970. For classical logic it has found its reflection in the deductive engine of the system for automated deduction SAD designed in the accordance with the EA requirements to automated theorem proving (see the Website "nevigal.org" as well as the papers [9–14]).

References

[1] G. Gentzen. *Untersuchungen uber das logische Schliessen.* Math. Zeit. (1934), 39, pp. 176–210.

[2] J.H. Gallier. *Logic for computer science: Foundations of automatic theorem proving.* Book. New York: Harper and Row, Inc., 1986, 513 pp.

[3] S. Kanger. *A simplified proof method for elementary logic.* In book: Computer Programming and Formal Systems. Studies in Logic and the Foundations of Mathematics. Amsterdam: North-Holland, Publ. Co., 1963, pp. 87–93.

[4] A.V. Lyaletski. *Variant of Herbrand theorem for formulas in prefix form* . Kibernetika (1981), 1, pp. 112–116. In Russian.

[5] J.A. Robinson. *A machine-oriented logic based on resolution principle.* J. of the ACM (1965), 12(1), pp. 23–41.

[6] A. Lyaletski and B. Konev. *Tableau method with free variables for intuitionistic logic.* Intelligent Information Processing and Web Mining (2006), pp. 153–162.

[7] L.A. Wallen. *Automated proof search in non-classical logics.* Book. Oxford University Press, 1990, 240 pp.

[8] O. Hiroakira. *Proof-theoretic methods in non-classical logic — an introduction.* In book: Theories of Types and Proofs. Mathematical Society of Japan, Tokyo, 1998, pp. 207–254.

[9] Yu. Kapitonova, A. Letichevsky, A. Lyaletski, and M. Morokhovets. *Algoritm Ochevidnosti - 2000 (a project).* Proc. of the 1st Int. Conf. UkrPROG'98, Kiev, Ukraine, 1998, pp. 68–70.

[10] A. Degtyarev, A. Lyaletski, and M. Morokhovets. *Evidence Algorithm and sequent logical inference search.* Lecture Notes in Artificial Intelligence, vol. 1705 (1999), pp. 99–117.

[11] A. Degtyarev, A. Lyaletski, and M. Morokhovets. *On the EA-Style integrated processing of self-contained mathematical texts.* In book: Symbolic Computation and Automated Reasoning: A K Peters, Ltd, USA, 2001, pp. 126–141.

[12] A. Lyaletsky, K. Verchinine, A. Degtyarev, and A. Paskevich. *System for Automated Deduction (SAD): Linguistic and deductive peculiarities.* Advances in Soft Computing: Intelligent Information Systems 2002, 2002, pp. 413–422.

[13] K. Verchinine, A. Lyaletski, and A. Paskevich. *System for Automated Deduction (SAD): A tool for proof verification.* Lecture Notes in Artificial Intelligence, 4603 (2007), pp. 398–403.

[14] A. Lyaletski and K. Verchinine. *Evidence Algorithm and System for Automated Deduction: A retrospective view.* AISC'10/MKM'10/Calculemus'10 Proceedings of the 10th ASIC and 9th MKM International Conference, and 17th Calculemus Conference on Intelligent Computer Mathematics, Paris, France, 2010, pp. 411–426.

Alexander Lyaletski Received July 28, 2015

Alexander Lyaletski
Taras Shevchenko National University of Kyiv
Address: Volodymyrska str., 64, 01601 Kyiv, Ukraine
Phone: (+38)(044)2293003
E–mail: lav@unicyb.kiev.ua

Part 5

Logics in informatics

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Universics: an Axiomatic Theory of Universes for the Foundations Part 1. Foundational Completeness

Ioachim Drugus

Abstract

This is the first part of a paper in 2 parts presenting an axiomatic theory called "universics" of mathematical structures called "universes" – structures like the "von Neumann universe", "Grothendieck universes" and the "universes of discourse" of axiomatic theories. Universics is pivoted around a "reduction principle" expressed as an axiom scheme and manifesting both as a generalized epsilon-induction principle in set theory and, dually, as a principle of deduction in logic. The methodology of universics is to discuss about the reality in terms of "universes" treated a special kind of structures, rather than to discuss about it in terms of "theories". As a motivation for this research served the Harvey Friedman's desideratum to develop a new foundational theory richer than set theory, which would be essentially based on a generalized induction principle and into which, partially or completely, could be immersed set theory. This desideratum emerged due to the "foundational incompleteness" of set theory – a property manifesting as impossibility to represent in its language all mathematical structures and concepts. The main result of this Part 1 is an "explication", i.e. a presentation in strict terms of universics, of the notion "foundational completeness". In Part 2 an algebraic set theory based on the ideas of universics is developed, which is believed to achieve the Friedman's desideratum.

Keywords: induction, deduction, reduction, well-foundedness, universics

1 Introduction

"Universics" is a term used by the author and, independently, by several philosophers, for a "theory of universes". The treatment of a phenomenon from the perspective of "the Universe", or from the point of view of "multitude of universes" idea is referenced (by philosophers) as "universic approach" – a term which has a meaning different from that of the term "universal". A term which sounds closely correlated with "universics" is "multiverse" – a term used to refer to the multitude of set theories, viewed from the perspective of their universes of discourse – a multitude, which emerges due to various methods of constructing universes, among which the best known is "forcing method" [1, 2]. The difference between two approaches is that universics is a theory of separate mathematical structures called "universes", while in the approach using the "multiverse", the universes are treated as parts of one large universe of discourse called "multiverse".

In this paper, universics obtains the features of a mathematical discipline with its specific methodology, which can be described as *direct discourse* about "universes of objects" versus the *indirect discourse* specific to the currently wide-spread methodology, which involves theories for which these universes are called "universes of discourse". Due to this methodology concepts like "universe of ideas", "universe of objects" or "universe of structures" emerge, which are structures without theories to describe them, and the "universe of theories" is the framework for explication of the notion "foundational completeness".

The "universes of sets" called "Grothendieck universes" were defined by Grothendieck as a foundation for category theory – a foundation which was needed, since the main concepts of this theory cannot be expressed in terms of the ZF set theory, or in terms of the von NBG (Neumann-Bernays-Godel) class theory – both previously considered "foundational". To be more specific, the "category of categories" is one of the key concepts of category theory, since "functor", "adjointness", "natural equivalence" and other central notions which determine the value of this theory, are defined proceeding from the idea that "category of categories" exists. But the existence of such an object presupposes

the existence of the "class of classes" – a concept which is "contradictory in itself", because such a class cannot be a member of any other class by the definition of "proper class". Moreover, category theory also uses such notions as "functor of functors", which imply existence of "proper classes of proper classes" – objects utterly unadmissible in any set theory or class theory. Thus, neither ZF nor NBG can be said to be really "foundational" for mathematics.

On the other hand, surprisingly or not, only one of the Grothendieck universes turns out to be sufficiently rich to serve as a "foundation for mathematics" – true, to the same extent as ZF, i.e. as an "incomplete foundation". This illustrates the necessity to have the concept "foundational completeness" for a theory, and this concept was introduced by Harvey Friedman in one of his messages in the automated email list FOM ("Foundations of Mathematics" https://cs.nyu.edu/pipermail/fom/1997-November/000143.html). As it was stated by Friedman, set theory "does not come close to doing everything one might demand of a foundation for mathematics" and, thus, it is not "foundationally complete" for mathematics.

In addition to the strong conceptual reasons mentioned above for the statement that set theory is foundationally incomplete for mathematics, one can also state that set theory is foundationally incomplete for informatics, since it is too poor for the representation of the data structures used in informatics. On the other hand, *no* arguments were found in favor of the statement that category theory, founded on Grothendieck universes, or just these universes may be foundationally incomplete. Therefore, a theory of universes, containing the Grothendieck universes, can be expected to be a foundationally complete theory. The main result of this Part 1 of the paper is the "explication", i.e. presentation in precise terms, of the notion "foundational completeness".

As an informal theory, universics was developed in several publications [3, 4], where universes were treated as the largest structures, similarly to how the proper classes called "universes" are treated in set theory. There are two main differences between set theory and universics – (1) set theory studies conceptions which are obtained by

abstraction from any kind of order, but the "universes" about which universics discusses are structures – universics is essentially "structuralist", (2) set theory studies "small scale" objects, but universics studies any structures, but could be said essentially to be about the "large scale" objects.

In previous publications on universics, the structures were considered as built by repetitive application of three operations, called "aggregation", "association", "atomification" – operations for building sets, ordered pairs and atoms, respectively. The reason for the choice of these notions as a starting point in building a foundational theory is the belief that the notions "set", "ordered pair" and "atom" are sufficient to serve as a "conceptual orthogonal basis" for a universe of concepts, where any concept can be reduced to this basis. This conceptuality is intended to describe the "fabric" of a universe, and could be called "small scale universics". The current paper presents a theory of structures called "universes", which can be called "large scale universics". Since the "fabric" of a universe is irrelevant here, any knowledge of those publications is not required for understanding the current paper.

2 On the terminology and conceptuality used in this paper

Universics is primarily intended to serve for the metamathematical analysis of various foundational theories. The terms used in a metadiscourse necessarily contain an amount of referential ambiguity and this section is intended to minimize such ambiguity for some terms. Also the conceptuality behind these terms will be clarified. In order that a term needing explanation can be easily found while reading the main text, the terms explained in this section are italicized.

A *universe of discourse* is correlated with a theory, which is said to "discuss" about the entities populating the universe, or to "describe" the universe. But there are clearly specified universes (like Grothendieck universes), for which no theory describing them have

been presented. Also, the von Neumann universe was invented as a view upon the totality of sets and only later it was found that ZF theory extended with terms for the ranks of the sets can describe it. This reasoning shows that the notion "universe" is to be treated as a notion on its own, prior to be correlated with a theory and to be called its "universe of discourse". Also, two theories may have the same universe of discourse and this fact also supports the treatment of universes as objects on their own.

The expression *universe of sets* is used alongside the expression "the universe of discourse of a set theory". Since in universics, the notion "universe" is treated on the same footing as the notion "theory", this practice will be preferred here, and expressions like *universe of ideas*, *universe of objects*, *universe of structures* will be used here, alongside the well-established term "universe of discourse".

Fraenkel used the term "object" for the entities in a universe. But since anything populating a universe is called "object", the expression "universe of objects" does not sound to be a good term. There is, though, a special case, when this term is the most appropriate one – this is when one opposes a "universe of ideas" to a "universe of objects" (to use the expression "universe of material things" instead of "universe of objects" would be incorrect, since an object in a "universe of objects" can also be an idea). Thus, the term *universe of objects* will be used here to express a meaning opposed to the meaning of the term "universe of ideas". The relationship between the elements of one of these universes with the elements of the other universe is called here *reflection* – more precisely, the universe of ideas is said to "reflect" the universe of objects.

An example of a "universe of objects" is the totality of all the things called "sets", "classes", "classes-as-many" – a term introduced by Russel to refer to collections which better reflect the plural of a noun than the notion of "class", "multi-sets" , or "aggregates" – a generic term introduced also by Russel for any kind of set-like objects. The algebraic "set theory" presented in Part 2 is weaker than the currently known (to the author) set theories and its objects are called "aggregations". This term was chosen to be close to "aggregates" to emphasize that these are

"generic set-like objects" like those of Russel, but it is different from "aggregates" to admit that they might be conceptions different from those ntended by Russel, also these objects are obtained by application of an operation of building sets called "aggregation".

The term *collection* is commonly used for an intuitive notion generalizing the notions of set and of class, without specifying whether or not a multi-set is a collection. Here a collection is presupposed to have no repeating elements, i.e. to be a set or a class. The exclusion of "multi-sets" from the denotata of the term "collection" does not mean that the multi-sets are excluded from universics – on the contrary, the set theory developed from the ideas of universics in Part 2 of this paper is essentially a multi-set theory. The exclusion of multi-sets from denotata of the term "collection" is just a convention about the use of the term "collection" in this paper.

The term "collection" is convenient in metamathematical analysis, especially, in discussions about the universes of discourse of set theories, where the notion of "size" (see below) is irrelevant to the topic of the discourse. In set theory, a set is treated as a class which is a member of another class. There are also classes which are not members of other classes, and these are called "proper classes". The modifiers "small" and "big" are applied to the noun "collection" to distinguish between sets and proper classes this manner: "small collection" is a synonym for "set", "big collection" is a synonym" for "proper class".

The modifiers "small" and "big" reference two values on a scale called "size" expressed in terms of the membership relation "$\in$" – a "big collection" is a maximal collection within the universe of collections governed by the membership relation, and a "small collection" is not maximal in this universe. This dimension can be also referenced as "height/depth" (depending on the perspective from which the membership relation is viewed). Accordingly, in category theory, a category with a big collection of objects and morphisms is said to be a "big category", and one with such a collection small, is said to be a "small category".

The universe of discourse of a set theory, let this be ZF, is a proper class (i.e. a "big collection"), but there are also other proper classes,

and a question arises: "what singularizes the universe of discourse of ZF among other proper classes and makes it a 'universe' other than being a 'universe *of discourse*'?". This question cannot be answered in terms of "size", as this term is treated today. Here, yet another dimension needs to be considered, which is referenced as "extension" – a collection C will be said to have a smaller extension than a collection D, if $C \subseteq D$. Notice, that in addition to being "big" in size, the universe of discourse of ZF also has the largest extension, and this answers the question regarding what singularizes the universe of discourse. To account for both "size" and "extension" dimensions, in universics the terms *small scale* and *large scale* will be used.

Finally, notice that the word "idea" is treated here as a term. Logicians consider the things used in a theory to be of two kindss – "notions" and "assertions" and they use the generic term *idea* for them (here, the "notions" can be – "properties", "relations", "functions", "operations", etc.). A theory is also an "idea" which can be treated as an inhabitant of a universe of ideas. If a universe of ideas is populated only by theories, the universe will be called "universe of theories".

3 What is a universe?

Since the notion of universe originates in logic, one can get a hint on what kind of mathematical structure is a universe exactly from logic, and namely from the definition of the notion "axiomatic theory", or shorter, "theory". Logicians define a theory as an entity which has a *basis* consisting of ideas of two sorts – "basic notions" and "axioms" (i.e. assertions, which are "postulated", or "posited in the basis"). Also, they consider the other ideas of the theory as obtained either by definition or deduction. To simplify the terminology, since axioms are "posited in the basis" of the theory, they can be called "basic assertions" (similar to "basic notions). Also, since the process of definition and the process of deduction are similar, the term *reduction* will be used for both these processes. According to this definition of the notion "theory", a theory uses two sorts of entities – notions and assertions. But one can admit that the notions are missing from a theory, and

thus, get to a "theory of assertions". Similarly, one can admit that the assertions are missing from the theory and get to a "theory of notions". Logicians do not use such theories, but nothing in the definition above prevent them from being "theories". Any of the one-sorted structures mentioned above is referenced here as a "universe of ideas".

Thus, the notion "universe of ideas" is defined as a triple consisting of a collection U, a subcollection B of U called *basis*, as well as a binary relation on U called *reduction*; U will be called here *support* of the universe. In this paper, the meaning of the symbol turnstile "⊢" used for deduction will be extended to refer also to notions and this symbol will be used for reduction of any ideas – assertions or notions. Thus, a "universe of ideas" is treated here as a simple mathematical structure denoted as a triple (U, B, "⊢").

Reduction is to be treated as a "generalization" of many relations on different types of objects. In previous sentence, the word "generalization" is used within quotation marks to emphasize that various other kinds of relations actually are not really "partial cases" of reduction – they are "reducible to reduction" (where the term "reducible" is to be considered metalinguistically, and the term "reduction" as pertaining to the language). This means that reduction is a fundamental binary relation and even "generalization", whatever is this, must be treated in terms of reduction.

The objects about which are the ideas in a "universe of ideas" are in a relationship of "reflection" with ideas, and here the supposition is made that this relation "projects" the same structure upon the objects, like in a homomorphic map (but reflection is a too complex phenomenon to be explicated as a homomorphic map). Therefore, a "universe of objects" will be considered here to have the same type of structure as a "universe of ideas".

A usable terminology and convenient notations are needed, and the development of these follows next. The symbol "⊣"symmetric to the symbol "⊢" will be also used, and the two expressions "x ⊢ y" and "y ⊣ x" will be referenced as two *presentations* of the same relation. Both expressions in the previous sentences a read the same: "x is reducible to y". Since, the order of arguments in this reading in a natural lan-

guage is the same as in the expression "x ⊣ y", it is natural to consider the expression "x ⊣ y" as the *direct* presentation of reduction, and the expression "x ⊢ y" as *inverse* presentation of reduction. In the definitions and proofs in this paper the direct presentation is preferred. There is no reason to consider one of these "presentations" as representing the "direct reduction *relation*" and the other – the "inverse reduction *relation*"; both will be considered as representing the same relation. Notice that also in logic, the "inverse relation" to deduction (treated as a relation) is rarely referenced, and it has no name. Still, an intuitive term can help in practice, and here the term "generate" and its derivatives is proposed for the relation inverse to reduction. Thus, the relation "x ⊢ y" can be read "x generates y".

A remark is in place regarding the treatment of reduction as a *binary* relation. Notice, that in case of deduction, in a correlation like "x ⊢ y", x is a list of assertions and y is one assertion – entities of two sorts. This forces considering both the lists of assertions and the assertions as entities of one sort – "idea". Similarly, a notion is generally reducible to a set of other notions and a "set of notions" is of a sort different from "one notion". This forces considering reduction of notions also as a relation between entities of the same sort – "ideas".

The reason why reduction is treated as a relation between two ideas, and not between one thing of a sort and many things of the same sort, is that the reduction relation *implicitly presupposes* the existence of a multitude of ideas to which one idea is reducible. It is exactly this implicit presupposition which made possible development of an alternative set theory to be presented in Part 2 of the this paper.

Similarly to ideas, *one* object is generally considered as reducible to *many* objects – say, a list of objects. The lists are structures – whether they contain objects or ideas. Also there are other types of structures about which one can say that they contain *many* objects. Therefore, the "universe of structures" is a type of universes in terms of which it is convenient to discuss about the properties of universes. Also, mathematics is about "mathematical structures", informatics is about "data structures" and the universes are defined as mathematical structures. This is favor of discussing about the universes in terms of

structures, while keeping in mind that the notions "universe of objects" and "universe of ideas" serve for ordering the discourse in a manner to take into account, when possible, which of these two kinds of structures intended to be a *reflection* of the other.

4 Universes of structures

For two structures x and y, the expression "x $\dashv$ y" is conveniently read as "x is a *reduct* of y". The term "reduct" used here comes from universal algebra, where an algebra A is said to be a reduct of an algebra B, if the signature of A is contained in the signature of B – a fact which can be imagined as "reducing" the B to A by "neglecting" or "ignoring" some of its operations. An intuitive synonym for "reduct" is "rudiment" or "rudimentary structure". In some cases, the expression "x $\dashv$ y" is conveniently read as "x is more elementary than y", and one of such cases is when x and y are sets – here, the relation of membership is a variety of reduction, and in set theory the correlation "x $\in$ y" is a case of reduction, which is read "x is an element of y" (which is a partial case of "x is more elementary than y") .

The collections can be treated as "final reducts" of the mathematical structures, since these are defined as having a collection as their support. Next, an explanation follows why the term "universe" is customarily used both for a collection and a structure which has this collection as its support. When the set theorists consider a universe of sets as an "internal model" of a set theory, no doubt they also take into account the membership relation which governs the sets in that universe – they treat such a "universe" as a structure. But they refer to such a structure as "class" making use of a linguistic device called "metonymy" – naming a whole by the name of a part. Thus, the reference to a structure by its support can be treated as a result of applying a "conceptual metonymy" device. In this particular case, the part of the structure used for reference is the "*final* reduct" of a structure, and this is an additional factor imposing to use the conceptual metonymy.

By applying the conceptual metonymy device, also other reducts of a universe are called universes. One of such reducts is of the kind

(U, "$\dashv$") obtainable by ignoring the basis. Such a universe will be said to be "baseless". The most representative example of a baseless universe used here is the universe (V, "$\in$") of discourse of the ZF, where "V" is the standard notation of the class of all pure sets (i.e. sets built out of the empty set, without using atoms). One cannot just "identify" (consider "the same") the pair (U, "$\dashv$") and the triple (U, $\emptyset$, "$\dashv$") by considering the same two different meanings – "*without* any base" (i.e. "baseless") and "*with* an empty base". Instead, one can use the conceptual metonymy device and call "universe", or more precisely "baseless universe", the pair (U, "$\dashv$") .

Generally, a pair (U, R), where U is a collection and R is a relation on U is said to be a "frame" – a term borrowed from non-classical logics, where this term is also used in a longer form – "Kripke frame". Thus, the frames will be treated here also as universes, baseless universes. Any relation is the relation within a frame, and therefore, in this paper, the symbols "$\dashv$" or "$\vdash$" be used for any relation.

There is yet another kind of reduct of a universe – a reduct obtained by discarding the reduction relation to obtain the ordered pair P = (U, B). Such a universe can be treated as a "problem" which is represented as U – the collection of "possible" solutions, and B – the set of "actual" solutions to the problem P. This type of "universes-problems" was proposed by Kolmogorov as an alternative interpretation of intuitionism. Finally, the reduct obtained by discarding both the reduction and the basis is a collection – thus, by using the conceptual metonymy device, the collections will be also referenced as "universes".

A proper definition of universes as structures – a definition accounting for the reducts of universes – *cannot* be done in the language of set theory other than by re-defining the notion of relation in a complicated manner. But such a "re-definition" can create risks of ontological and terminological inconsistency. There is, though, an approach, which offers a convenient device for the presentation of universes as structures – the approach presented in [5] which uses the notion "quasiary relation". Roughly, a quasiary relation is a relation with optional correlates. Also, there is a mereological set theory [6] with the "empty set" treated as "nothing", where the pair (U, "$\dashv$") and the triple (U, $\emptyset$, "$\dashv$") can be

treated as two notations of the same thing. Finally, the universes can be conveniently represented in "aggregation theory" presented in Part 2. It also sounds plausible, that the notion of "conceptual metonymy" can be explicated in all these three approaches.

Another important question is whether other types of structures (which "look differently") are reducible to the structures of type "universe". This aspect was not researched in detail by the author, but there is a result of Quine (seemingly, one which practically did not draw any attention of the mathematicians). Namely, Quine showed that the combinatory logic of Moses Schoenfinkel can be interpreted as a logic of relations (rather than functions) [7]. This result of Quine can serve as a basis for the belief, that all possible kinds of structures can be represented as universes.

5 Definition of universes as structures and related basic notions

A subcollection X of a frame (U, "$\dashv$") is called *transitive* in the frame, if the following condition holds:

$$(\forall u,\, v \in U)\ ((u \dashv v)\ \&\ v \in X \rightarrow u \in X).$$

The intersection of all transitive subcollections of U comprising the collection X is called transitive closure of X and is denoted as "[X]".

A universe was said to be a structure of the type (U, B, "$\dashv$"), or of the type (U, B, "$\dashv$"), but nothing was mentioned about the properties of such a structure, which are *mandatory* for them to be "universes".

The strict and (complete) definition of the notion "universe" is the one below.

Definition. A triple (U, B, "$\dashv$"), where U is a collection, "$\dashv$" is a binary relation on U, and B is a transitive subcollection of the frame (U, "$\dashv$"), is called a *universe* with the *support* U, *basis* (or, synonymously *foundation*) B, *frame* (U, "$\dashv$"), *reduction relation* "$\dashv$", and *co-basis* U\B.

Thus, the mandatory property of a structure of the type (U, B, "$\dashv$") is the transitivity of the basis B – a property required by the

meaning of the terms "basis" (or "foundation"), which does not allow that "below" an element in the "basis" (or "foundation") there can occur any elements of the co-basis.

The co-universe of a universe (U, B, "$\dashv$") is defined as the universe (U^c, B^c, "$\dashv^c$"), where $U^c = U$, $B^c = U\backslash B$ (where "$\backslash$" denotes the set-theoretic difference), and "$\dashv^c$" graphically coincides with "$\vdash$". The universe and its co-universe are said to be "dual" to each other. Notice, that a universe dual to a universe U is not just a universe with the symmetric presentation, but also with the co-basis of U as its basis.

In a customary manner (as for example, in category theory), a "dual notion" with prefix "co-" added to its name is defined for each notion. The superscript "c" will be used in the denotation of a dual notion as above in the definition of the notion "co-universe" or, for example, in notation $[X]^c$ for the co-transitive closure of a subcollection X. Obviously, the term "co-basis" was used in the definition of "universe" for the collection $U\backslash B$ specifically because this is a dual notion to the notion "basis". The co-basis is a co-transitive subcollection of a universe.

Obviously, $[B]^c = [B^c]$, which in words sounds like this: "the co-transitive closure of the basis is equal to the transitive closure of the co-basis". Therefore, the two conditions $[B]^c = U$ and $[B^c] = U$ are equivalent. In words, the first condition means: "the basis must generate the universe". This condition on the structure called "universe" seems to be mandatory, because mathematicians often treat the notion "basis" as in the terms "generating basis" or "basis of generators" and they use modifiers like "finitely-based". The term "foundation" does not have this connotation of "generation" and the "foundation" could have been preferred for the component "B" of a universe. Unfortunately, the term "induction basis" is widely accepted and cannot be easily replaced with the term "foundation of induction". Therefore, the term "basis" will be preserved, but one needs to remember that this term is not supposed to have the connotation of "generation". A more important note is that, in order to allow for a wide class of universes, where in particular, the notion "foundational completeness" makes sense, the property of universes to be generated by the basis will

not be required. The universes generated by their bases will represent an important class of universes said to be "well-founded".

The notions as direct product, homomorphich image, etc. are defined in the customary manner and will not be introduced here. The only notion in addition to those already introduced which is needed here is that of a "sub-universe" and it will be defined here like this: a universe U is said to be a sub-universe of the universe V, if the support, basis, and the reduction relationship of U (treated as a collection of ordered pairs) are included in the support, basis and reduction relationship of V, respectively.

In this section, the notion "universe" was presented in terms of relations – a unary relation, a "property" and a binary relation. In section 11, an algebraic presentation of this notion will be introduced, where a universe will be treated as a universal algebra.

6 Universics as a framework of axiomatic theories

The term "set theory" refers to the conception of set and to a large number of axiomatic set theories, which are correlated with each other, since all these theories describe different aspects of the same conception. Thus, set theory is a framework of axiomatic set theories. Also, this "theory of universes" or "universe theory" (similar to "theory of sets" but "set theory"), is a framework of axiomatic theories. These theories use the 1^{st} order predicate language, with a unary predicate symbol "B" with the meaning of B(x) expressed in words like "x is basic", or "x is fundamental" or "x is foundational" (depending on the particular universe), and the binary predicate symbol "$\dashv$" to be used as infix, with the predicate symbol "$\vdash$" used "symmetrically".

Another notation for B can be the symbol of reduction "$\dashv$" used as a unary predicate symbol in prostfix position as in expression "x $\dashv$ ", and similarly for the symbol "$\vdash$" to be used in a prefix position, and alternatively, functional notations "$\dashv$(x)" and "$\vdash$(x)" can be used. But this would only reduce the number of symbols of the language to one,

while impeding to the "readability" of formulas. Instead, this use of the symbol of reduction for a unary predicate symbol shows that the property "to be basic" can be intuitively treated as a rudiment of the relation "to be reducible". Notice, that instead of the two symbols "B" and "⊣", only the symbol "⊣" can be used, if reduction is considered a symbol of a quasiary relation [5]. Thus, universics can be treated essentially as a theory of one quasiary relation. The described language will be called "language of reduction" or "reduction language". This language can be extended by adding new symbols, and to oppose it to any extended language one can use the modifier "pure" like this: "pure reduction language".

Since the universics is essentially a theory of large scale universes which serve as universes of discourse of theories, this theory must explicate the most fundamental properties like "to be fundamental", "to be foundationally complete" or "to be well-founded". "To explicate" is also to present the properties or properties like those mentioned in previous sentence. A better term than "property of properties" is the term "principle" and the postulates of universics are called "principles" – "universic principles". There are two such principles – the "fundamentality principle" (1) and the "reduction principle" (2), which are presented below:

$$(\forall\, x, y)\ ((x\ \dashv\ y)\ \&\ B(y)\ \rightarrow\ B(x)), \tag{1}$$

$$(\forall x(B(x) \rightarrow P(x))\ \&\ \forall x(\forall y((y\ \dashv\ x) \rightarrow P(x)) \rightarrow P(x))) \rightarrow \forall x P(x). \tag{2}$$

The principle (1) is an axiom, and the principle (2) is an axiom scheme, where P is a formula and "x" is a variable which may or may not enter in P, together forming a property. These two principles are the axioms of a theory denoted here as **W** – a theory of "well-founded universes". The principle (1) mainly explains the meaning of the symbol "B" and could be added to the conjunction in the antecedent of the principle (2), so that this would describe the properties of B(x) and of P(x) and provide the condition for the conclusion to hold. Thus, the theory **W** can be said to be based on the reduction principle modified in this manner. But (1) and (2) are independent, and a theory based

on only (1) offers the facilities to deal with the conception of fundamentality and foundational completeness. Therefore, a theory denoted as "**U**" with only the fundamentality principle (1) is worth considering. The meaning of these two postulates will become clear in the course of this presentation where the universes of discourse of these two theories are described to provide the semantics the **W** theory.

7 On the ontology of universics

Ontology is the science of "being" – philosophically, it is a science about existence; conceptually, it is about what something "is" – something can be an atom, a set, a structure; and linguistically, it manifests in terms ("atom", "set", "structure") which form "thesauri" – linguistic counterparts of "ontologies".

As mathematical structures, the universes serve as *explications* of phenomena – their representation as abstract structures called "universes". Due to the fact that different concrete phenomena can be explicated as one structure, certain particularities of those phenomena are ignored, which is presupposed to occur in any process called "abstraction". Other particularities of phenomena are essential and they are reflected in the terminology used for the explication. Sometimes, different terms used in discourse about the original phenomena reflect the same feature in their explication. This can be exemplified by three universes – universe of structures, universe of notions, universe of theories. In the universe of structures, the basis is a collection of structures called "basic". In the universe of notions, the basis is a collection of notions called "fundamental". In the universe of theories, the basis is a collection of theories called "foundational". All these three universes are the same as mathematical structures, but the three notions are called differently: "basic", "fundamental", "foundational".

Since universics is also a "universal theory", the varieties of phenomena which can be explicated as the same structure must be huge and the terminology used for describing them – "variegated". Thus, variants of a term must be admitted in a definition, and the same notion must be recognized in different system of terms (thesaurus). Among

the best tool for orientation in this conceptual and terminologic wealth is a proper handling of presentations of the universes.

8 Two presentations of a universe

Even though a universe of ideas and a universe of objects are explicated as structures of the same type, the practice of using structures of this type shows that the convenient presentations of the two universes differ – these are symmetric diagrammatic representations. So, while a universe of ideas is conveniently presented as a triple like this (U, B, "$\vdash$"), a universe of objects is more convenient to be presented as a triple like this (U, B, "$\dashv$") with a symmetric symbol for reduction. There is yet another kind of "inversion" – while the presentation of reduction relation for ideas is convenient to be considered as "inverse", the presentation of reduction for objects is convenient to be considered as "direct". There are yet more "inversions" when the objects or the ideas are more particular, and such "symmetric projections" are due to the complex laws of reflection relation between ideas and objects.

The choice of a presentation for a universe representing a phenomenon is important for handling the terminology, conceptuality and the development of the intuitions related with explicated phenomenon and the next two examples illustrate this.

Example 1. Notice that the objects called "structures" are intuitively treated as obtained in result of a "construction" process which unfolds in "steps". A set is a particular case of structures – a set is constructed from its elements by repetitive application of an operation – one which is called here "aggregation". In the description of the von Neumann universe, the "steps" are called "stages", and a set is obtained at a certain stage of this process. This kind of "construction" process is treated here as a special kind of reduction, when they consider that "x is a reduct of y", if x is in the relation $[\in]$ with y, where $[\in]$ is the transitive closure of the membership relation.

Any process of construction in "steps" or "stages" is intuitively perceived as ruled by the mathematical induction principle, and one can say that induction is the explication of the construction process. Prob-

ably, Harvey Friedman was guided by this idea, when he considered a set as obtained in result of an "inductive construction". Based on these intuitions the reduction relation of the universe of structures is said here to be of "inductive type".

Example 2. In a universe of ideas, an idea x cannot be said to be "constructed" from another idea y, since each idea "reflects" other objects, and the ideas are created in mind to comply with the laws of "reflection" of reality – not in compliance with the relations between the reflected objects . A universe of ideas is many-sorted, and the term for reduction is sort-specific. As earlier discussed, there are two main sorts of ideas which are mostly treated by logicians – assertions and notions. For two assertions x and y, one says that x is *deducible* to y, and denotes this as "$x \vdash y$". The deduction relation is another case of reduction. The axiom "$x \vdash x$" is postulated, practically, for all deduction calculi. But if this axiom is not postulated, a generalization of deduction is obtained, and the features of this generalized deduction can be described by a principle dual to induction principle – one which is appropriate to be called "deduction principle". The deduction calculi define various universes, and all of them are said to be of "deduction type".

For each notion in the presentation of one type, there is a dual notion in the other presentation of the other type which will be called here "dual image" of the notion. This invites for the search of dual images in the universes with dual presentations.

9 Atoms as irreducible objects of a universe

For a set theory to have atoms, its language must have a predicate symbol, usually denoted as Atom(x) in addition to the membership symbol. This predicate has the meaning "x is an atom", and those objects for which this predicate is true are "called atoms". The atoms make up a set (not a class). The universe of discourse of a set theory with atoms (U, B, "$\in$"), where B is the set of atoms, is a "full-fledged" universe – it is not a reduct, like the universe of ZF. Here, by "atoms" the "regular atoms" are referenced – objects, which do not have ele-

ments, and which are different from the empty set – an object called also "urelements". The urelements are objects of a sort different from the sort "set".

A "Quine atom" is an object q which equals to its singleton {q}, or in other words, a Quine atom is a set (!) in which membership relation is reflexive. Thus, a Quine atom is not a proper "atom" – it is a singleton, a special kind of set. Hence, a set theory whose all atoms are Quine atoms is a "pure set theory". If q is a Quine atom in a set theory, then there exists an infinite chain in the universe of discourse of this theory: $q\in q\in \ldots$. A Quine atom is a non-well-founded object, and the transfinite induction principle cannot be proved in a set theory with Quine atoms.

By analogy with set theory, two formulas are introduced here for universes: $\mathrm{Urelement}(x) \stackrel{\mathrm{def}}{=} \neg\exists y(y \dashv x)$ and $\mathrm{QuineAtom}(x) \stackrel{\mathrm{def}}{=} \forall x((y \dashv x) \leftrightarrow y = x)$. Notice, that these two formulas express different kinds of irreducibility. An object of a universe with the property Urelement(x) is called "urelement", and an object of a universe satisfying the condition QuineAtom(x) is called "Quine atom". The property to be an "atom", i.e. either an "urelements" or a "Quine atoms" is defined like this:

$$\mathrm{Atom}(x) \stackrel{\mathrm{def}}{=} \quad \forall y((y \dashv x) \rightarrow y = x).$$

If the basis of a universe consists only of atoms, the elements of the basis are pairwise incomparable, i.e. the basis satisfies the following condition: $(\forall x, y \in B)((x \dashv y) \,\&\, (y \dashv x) \rightarrow x = y)$.

Call a universe "atom-based" if all objects in its basis are atoms, and call the basis of such a universe – "orthogonal basis" of the universe.

10 What is foundational completeness?

Despite its extremely "weak" character of a theory which imposes only one condition on universes – to satisfy the "principle of fundamentality", the theory **U** can be used to explain many notions, like "fundamental idea" or "foundational completeness of a theory". Obviously, the most appropriate universe for treatment of the first notion is the "universe of ideas" and a convenient presentation for this universe is

the deductive type presentation. Thus, the convenient symbol for reduction of ideas is the symbol "$\vdash$". The expression "$x \vdash y$" will be read here as "x is more fundamental than y", and the formula "$B(x)$" as "x is fundamental".

U has only one axiom called "fundamentality principle", which in this deductive type presentation looks like this:

$$(\forall x, y)\ ((x \vdash y)\ \&\ B(y) \rightarrow B(x))$$

The intuitive meaning of this axiom is expressed in this reading of the fundamentality principle: "if an idea is fundamental, then a more fundamental idea than it is also fundamental".

This principle makes little sense for axiomatic theories, but it makes a lot of sense for theories which are not axiomatic. So, the intuitive set theory is not an axiomatic theory – it is a universe of statements among which some statements are considered as mandatory for describing the conception "set" and are declared to be fundamental. If a statement B is considered fundamental, and later another statement A was found, such that $A \vdash B$, then the fundamentality principle prescribes to consider "fundamental" also the statement A.

An axiomatic theory T has a basis, where its axioms are contained, which is a sub-universe of the universe of intuive theory axiomatized by T. An idea can be fundamental in the universe of the intuitive theory, but not "posited in the basis of T", i.e not declared as an axiom. If it is posited in the basis of the theory T, then it becomes an axiom of the theory T.

The notion "axiom" of a theory is similar to the notion "atom" in the dual universe or, in other terms – it is a "dual image" of the notion "atom". There is no short term to a theory with independent axioms and such a theory will be said here to be "orthogonally-based" similar to the dual notion used for the universe of set theory. It makes sense to consider only such theories for definition of the notion "foundational completeness" below.

Definition. Suppose (T, A, "$\vdash$") is an orthogonally-based theory with A as its set of axioms, and T is a sub-universe of a universe (U,

B, "$\vdash$"). Then the theory T is said to be *foundationally complete* for the universe U, if [A] = B.

For U, a well-founded universe, the equality [A] = U is true, but using this condition in the definition would limit it to only well-founded universes. As it was formulated, the definition provides the most large application of this notion, including, to the non-well-founded universes.

11 A general method for algebraization

In this section, three general methods for representing a relation via an operation are introduced which can serve for algebraization of universics. Actually, due to [7], these methods can be used for algebraization of any theories, including those with symbols for n-ary relations, where $n > 2$.

Recall that a collection equipped with a binary operation "$\circ$" is called "magma" or "grouppoid". But the term "grouppoid" is also used with another meaning in category theory, and here the term "magma" is preferred. For a magma M = (S, "$\circ$"), the collection S is called "support of the magma" and is often denoted as $\underline{M}$, while "$\circ$" is the symbol of an operation, called "(fundamental) operation of magma". The methods of algebraization presented in this section are for algebraization of a frame, a magma is associated with any frame, such that the frame can be restored from the magma. Since any frame can be considered as a baseless universe, the fundamental relation of a frame will be denoted here as "$\dashv$".

Method 1. This method of algebraization presupposes that, with any magma M = ($\underline{M}$, "$\circ$") the frame F = ($\underline{M}$, "$\dashv_\circ$") is associated, where the relation "$\dashv_\circ$" is defined in this manner:

$$x \dashv_\circ y \text{ iff } x \circ y = y.$$

A mathematical structure, in particular, a magma or a frame, is called trivial if it has only one element. It is easy to prove the following proposition.

Proposition. Any non-trivial frame associates with a non-trivial magma.

Proof. Suppose F = ($\underline{F}$, "$\dashv$") is a frame, and define the operation

"$\circ_{\dashv}$" in the manner showed below, where x and y are arbitrary elements of the frame:

(1) if $x \dashv y$ and $x \neq y$, then set $x \circ_{\dashv} y = y$;

(2) if $x \dashv y$ does not hold and $x \neq y$, then $x \circ_{\dashv} y = x$;

(3) if $x \dashv y$ holds and $x = y$, then $x \circ_{\dashv} y = z$, where z is any element different from x (such an element always exists, given that the frame is non-trivial, i.e. it has at least two elements).

It is easy to check that the relation "$\dashv$" coincides with the relation "$\dashv_{\circ}$", and if this relation is anti-reflexive, i.e., if $\neg(\exists x \in S)(x \dashv x)$, then the operation, with which the frame F is associated is defined only by the conditions (1) and (2).

This Method 1 will be used in Part 2 for development of an algebraic set theory based on the ideas of universics.

Method 2. Any binary relation "$\dashv$" can be treated as obtainable from an operation "$\circ$" in this manner: $x \dashv y$ iff $\exists u(x \circ u = y)$. This method can be used for algebraization of a theory of sequences as this will be shown in Part 2.

Method 3. This method of algebraization of theories is based on the idea to use the "graph algebras" [8]. Basically, a "directed graph" (or "oriented graph" in terminology of other authors) is a frame F and its "graph algebra" is defined as a magma $(\underline{F} \bigcup \{0\}, \text{"}\circ\text{"})$, where "0" denotes an entity which is not a member of $\underline{F}$, while the operation denoted as "$\circ$" is such that $x \circ y = y$, if $x \dashv y$, and $x \circ y = 0$, in all other cases. There is no agreement regarding (a) the denotation "0" – sometimes one uses the denotation "∞" instead, (b) whether the operation "$\circ$" is to be defined like above, or it is appropriate to define it so that $x \circ y = x$, if $x \dashv y$, (c) to consider or not the symbol "0" a symbol of a 0-ary operation. This lack of an agreement is beneficial for algebraization purposes using graph algebras, since the appropriate variant can be chosen based on the concrete theory which "undergoes" algebraization. No attempt was made by the author to algebraize universics by means of graph algebras, even though this approach sounds

to be very useful for development of an algebraic set theory based on the ideas of universics, if the element denoted as "0" is treated as the empty set.

A consistent practice to denote the binary operations associated with binary relations is adopted here. The operation associated with the relation denoted as "$\dashv$" is denoted as "$\cdot\!:$" and the operation associated with the relation "$\vdash$" is denoted as "$:\!\cdot$". A mnemonics for the operations similar to the mnemonics for the relations is adopted – "one dot" is directed towards the entity treated as "one" and "two dots" (the colon) is directed towards the entity treated as "many".

In conclusion of this section, a discussion is in place about the comparative power between a theory and its algebraic counterpart, which relates also to methodology of presentation of theories. A binary operation "$\circ$" on a collection C can be represented as a collection of triples (x, y, x $\circ$ y) and, thus, it can be treated as a partial case of a ternary relation, and a ternary operation can be said to "carry more information" than a binary relation treated as a collection of ordered pairs. This additional information encrypted in binary operations allows for more expressivity. So, while the notion of function needs to be defined in a set theory based on the membership relation, in algebraic set theory, a function can be defined via the fundamental operation which represents the membership. Also, the generalized induction principle in the algebraic set theory permits to naturally define the (transfinite) recursive functions.

References

[1] J. Vaananen. *Multiverse set theories and absolutely undecidable propositions.* In: Interpreting Goedel, Critical Essays. ed. J. Kennedy (2014) pp. 180–205.

[2] J. D. Hamkins. *The set-theoretic multiverse: A natural context for set theory.* Annals of the Japan Association for Philosophy of Science (2011) vol. 19. pp. 37–55.

[3] I. Drugus. *Universics: a Common Formalization Framework for Brain Informatics and Semantic Web.* In: Web Intelligence and Intelligent Agents. InTech Publishers, Vucovar (2010), pp. 55–78.

[4] I. Drugus. *Universics: an Approach to Knowledge based on Set theory.* In: Knowledge Engineering Principles and Techniques. Selected Extended Papers. Cluj-Napoca, Romania (2009), pp. 193–200.

[5] M. S. Nikitchenko, S. S. Shkilniak. Algebras of quasiary relations. Theoretical and Applied Aspects of Program Systems Development (TAAPSD'2014): 11th international conference: proceeding. K., 2014. pp. 174–181.

[6] D. Lewis. *Parts of classes.* Basil Blackwell. 1991. p. 4.

[7] W. V. Quine. *A reinterpretation of Schönfinkel's logical operators.* Bull. Amer. Math. Soc. Volume 42, Number 2 (1936), pp. 87–89.

[8] G. F. McNulty, C. R. Shallon. *Inherently nonfinitely based finite algebras*, Universal algebra and lattice theory (Puebla, 1982). Lecture Notes in Math. 1004, Berlin, New York: Springer-Verlag, 1983. pp. 206–231.

Ioachim Drugus

Received August 2, 2015

Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
5 Academiei str., Chisinau, MD-2028, Moldova
E–mail: ioachim.drugus@math.md

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Universics: an Axiomatic Theory of Universes for the Foundations Part 2. Well-Founded Universes and An Algebraic Set Theory Based on Universics

Ioachim Drugus

Abstract

This is the Part 2 of the paper about an axiomatic theory of universes called "universics". Two main results are presented here: (1) a generalization of the notion "well-founded set" to the notion "well-founded universe" and the proof of a theorem saying that the theory W introduced in Part 1 axiomatizes the class of well-founded universes, and (2) an algebraic set theory based on the ideas of universics, into which set theory can be immersed. This is contended to achieve the Harvey Friedman desideratum to find an alternative theory pivoted around induction, into which set theory or a large part of it could be immersed.

Keywords: well-founded universe, Noetherian universe, multi-identity object, aggregation.

1 What is a well-founded universe?

The property to be well-founded was initially defined for sets [1], but it can be extended to arbitrary collections in this manner: a collection U is called *well-founded*, if for any non-empty sub-collection V of U, there is a "minimal" element m such that for any $v \in V$, it is false that v is in relation $[\in]$ with m (here $[\in]$ is the transitive closure of the membership relation). Frequently, the property to be well-founded is defined for binary relations, which is a generalization of this notion,

but for this presentation, the initial meaning of the term "well-founded" is important. The support of a universe can be well-founded, but to define a universe as "well-founded" only because of it having a well-founded support, would mean to ignore the fact that a universe also has a basis. A better definition would be one, which takes into account also the basis, but allows for a well-founded universe to contain non-well-founded objects.

In a set theory with the axiom of choice, it can be proven that a set s is well-founded, iff there does not exist an infinite chain s_0, s_1, s_2,. . . , where $s = s_0$, and for any n = 0, 1,. . . , $s_{n+1} \in s_n$ – a condition called "decreasing chain condition" in ring theory, and which is satisfied by the Noetherian rings. Thus, in a set theory with the axiom of choice, well-founded-ness of sets is equivalent to the decreasing chain condition – a condition which can be used in definition of well-founded-ness of a universe. In this paper, a universe will be considered to be well-founded if any infinitely decreasing chain is contained in its basis starting from a certain point. This definition does not exclude any non-well-founded sets in a universe of sets and allows for a very large class of universes.

2 The meaning of the reduction principle in universics

The meaning of the reduction principle can be better understood if it is explained in terms of the mathematical induction principle in arithmetic where induction has a basis (usually "1"), and in terms of epsilon-induction principle in set theory, whose universe of discourse is a baseless universe, and where induction does not have a basis as this was noticed in the early days of set theory [2]. In set theory, the "epsilon-induction principle" is the following scheme of statements in the language of set theory:

$$\forall x(\forall y(y \in x \to P(y)) \to P(x)) \to \quad \forall x P(x).$$

Here, "P(x)" is an arbitrary "property of x", with "P" a formula and "x" – a variable which is not required to necessarily occur in formula P. This principle is equivalent to the regularity axiom, given also the

other ZF axioms.

A generalization of epsilon-induction is, obviously, obtained by using an arbitrary binary relation R on a class X instead of the membership relation. Such a generalized induction holds for any "well-founded" relation R – i.e. a relation R, where any non-empty subset S of class X has a "minimal" element, i.e. an element s, such that (x R s) does not hold for any $x \in X$.

The epsilon-induction does not hold in a set theory with Quine atoms. More generally, it does not hold in any set theory with non-well-founded sets. Naturally, the idea comes to mind to "hide" all such sets in a basis of induction, and reformulate the epsilon-induction to a principle which holds in any set theory, including in set theories with non-well-founded sets. Based on this idea, the epsilon-induction principle can be generalized and presented like this:

$$(\forall x(B(x) \to P(x)) \ \& \ \forall x(\forall y((y \in x) \to P(y)) \to P(x))) \to \forall x P(x). \quad (1)$$

This epsilon-induction principle is a principle of induction presented in the most "general form" and if the symbol "$\in$" is replaced with the symbol "$\dashv$", the "reduction principle" of universics is obtained in its direct presentation, which can be called (just) "induction principle". If the symbol "$\vdash$" is used, then the following form of the reduction principle, which can be called "deduction principle", is obtained:

$$(\forall x(B(x) \to P(x)) \ \& \ \forall x(\forall y((y \vdash x) \to P(y)) \to P(x))) \to \forall x P(x). \quad (2)$$

If the property P(x) is a property of assertions and the expression "P(x)" has the meaning "x is true", then the deduction principle states that, if the axioms of a theory are true, and if the truthfulness of an assertion follows from the truthfulness of all assertions from which it can be deduced, then all the assertions of the theory are true. All deduction calculi are built in such a manner that this principle is satisfied and this principle can be treated as a "metamathical law" of deduction.

The reduction principle of universics does not assign any of the two meanings discussed above, that of induction or that deduction, and

it could be interpreted in many other manners different from those above. For the practice of using the reduction principle, several terms are useful and these will be introduced next. The formula ($\forall$x(B(x) $\rightarrow$ P(x)) is called "basis condition" and the formula $\forall$x($\forall$y((y $\dashv$ x) $\rightarrow$ P(x)) $\rightarrow$ P(x))) is called "reduction condition". A property P(x) satisfying the reduction condition is called "reductive property". A property P(x) is called "universal" if the formula $\forall$xP(x) is true. Thus, the reduction principle can be read in this manner: "if the elements of the basis of a universe have a reductive property, then this property is universal".

3 Well-founded universes

Denote the set of natural numbers as "N" and call a sequence u_n, n $\in$ N, of elements of U *decreasing* (*increasing*) *chain* in universe U, if for any n, the condition $u_n \dashv u_{n+1}$ (respectively, $u_n \vdash u_{n+1}$) holds, and say that such a chain *starts in* u_1. Even though the modifier "*infinite*" will not be used, such a chain will be considered infinite due to the infinite set of indexes (N). Thus, the chain u $\vdash$ u $\vdash$ . . . , for some u is considered as infinite. Such a chain occurs in any universe U, in which there exists an element u, such that u $\vdash$ u. Notice, that a finite sequence $u_1 \vdash \ldots \vdash u_k$ cannot be said to be "everywhere" decreasing, because one cannot say that it also "decreases in u_k", once there is no u_{k+1} such that $u_k \vdash u_{k+1}$. Due to this intuition, the modifiers "decreasing" and "increasing" are treated here as implying also the modifier "infinite" and the expression "infinite decreasing (increasing) chain" will not be used.

Call *well-founded* (*co-well-founded*) universe a universe U, such that for any decreasing (increasing) chain u_n, n$\in$ N, there exists a k $\in$ N, such that $u_k \in$ B. A baseless well-founded (co-well-founded) universe is called "Noetherian" ("Artinian"). An example of a Noetherian universe is the universe of ZF, and an example of an Artinian universe is the universe of all hereditarily finite sets. A set s is hereditary finite if both s and any element of the transitive closure [s] is finite (an example of a non hereditary finite set is the set $\{\omega\}$, where "ω" is the first infinite

ordinal).

This treatment of well-founded-ness in terms of decreasing (increasing) chain condition is similar to the approaches used by Emmy Noether and Emil Artin in ring theory, except that in universics this condition is modified to take into account also the basis of the universe. One may wonder whether the term "well-founded" (in particular, "Noetherian") and "co-well-founded" (in particular, "Artinian"), which are defined in terms of decreasing, respectively – increasing, chain conditions, are properly defined with respect to presentation – "direct" versus "indirect".

Actually, the choice of the terms "well-founded" (or "Noetherian") and "co-well-founded" (or "Artinian") is appropriate, as it follows from next simple proposition:

Proposition. For a decreasing (increasing) chain u_n, $n \in N$, of a universe U, the sequence $[u_n]$, $n \in N$, where for any $n \in N$, $[u_n]^c$ is a transitive (co-transitive) closure of u_n, is increasing (decreasing).

Now, to make sure that the presentation was chosen properly, notice that if a chain of elements of a universe is "increasing" the corresponding chain of ideals is "decreasing", and viceversa. Also notice that in ring theory, the notion Noetherian (Artin) ring is defined in terms of decreasing (increasing) chain of ideals of the ring and not of elements of the universe like in case of universes.

For any decreasing (or increasing) chain u_n, $n \in N$, call *tail-chain* (or simply *tail*) of it any decreasing (respectively, increasing) chain v_n, $n \in N$ such that there exists some $k \in N$, so that for any $n \in N$, $v_n = u_{n+k}$. Since the base B of U is a transitive collection, for each decreasing chain there is a tail of it which is entirely in B.

The statement below is a lemma for the Theorem which follows it.

Lemma. Any non-empty sub-collection V of the co-basis of a well-founded universe is well-founded.

Proof. Notice that the meaning of the term "well-founded" is different for the sub-collection and for the universe – a well-founded collection is a collection such that any sub-collection of it has a minimal element as this was explained in section 1. Suppose, there are no minimal elements in V and, since V is a non-empty collection, choose an

arbitrary $u_1 \in V$. Since u_1 is not minimal, there exists a $u_2 \in V$, such that $u_1 \vdash u_2$, and so on. Reasoning this manner, one can build a decreasing chain $u_1 \vdash u_2 \vdash \ldots$ of elements of V, and since $V \subseteq U \backslash B$, there is no tail-chain of this chain in B. This contradiction with the fact that U is a well-founded universe shows that the supposition that V is well-founded is false, and thus, that the proposition is true. QED.

Recall that **W** is the theory with both the principle of fundamentality and the principle of reduction.

Theorem. A universe U is a model of **W** iff U is well-founded.

Proof. Let U be a well-founded universe and prove that U is a model of **W**. Suppose, that P(x) is an arbitrary property, and prove the reduction principle for P(x). Suppose, the condition of reduction principle is true, i.e. the assertion (*) holds.

$$\forall x(B(x) \to P(x)) \ \& \ \forall x(\forall u(u \dashv x \to P(u)) \to P(x)) \qquad (*)$$

Denote by V the sub-collection of all x in U which do not have the property P(x) , and suppose that V is non-empty, and that there exists a $u \in V$. Since $\forall x(B(x) \to P(x))$ but P(u) does not hold, according to the definition of V, one gets that B(u) also does not hold. Therefore, $u \in U \setminus B$. Thus, it was shown that $V \subseteq U \backslash B$. Due to the Lemma, there is at least one minimal element in V and denote as m one of such minimal elements. Since m is minimal, $\forall u(u \dashv m \to u \notin V)$ is true and, since $u \notin V$ means exactly that P(u) holds, one gets that $\forall u(u \dashv m \to P(u))$ holds. Due to (*), $\forall u(u \dashv m \to P(u)) \to P(m)$. Thus, P(m) holds, and this contradicts to the supposition that $u \in V$. This concludes the proof that any well-founded universe is a model of theory **W**.

Now, to prove the contraposition, suppose that a universe U is not well-founded, and that there exists a decreasing chain $u_1 \vdash u_2 \vdash \ldots$ in U, such that no tail of it lies in B (notice that now the inverse presentation symbol "$\vdash$" is used). Since B is a transitive collection, for all $n \in N$, $u_n \in U \backslash B$, i.e. this chain lies in the co-basis. Denote as "P(x)" the property "$(\forall n \in N)(x \neq u_n)$" and show that the reduction principle for P(x) is a false statement. Namely, show that the following three statements are true: (a) $\forall x(B(x) \to P(x))$, (b) $\forall x(\forall u(u \dashv x \to P(u)) \to P(x))$, (c) $\exists x \neg P(x)$. To show (a), suppose u is an arbitrary

element of B and notice that the chain $u_1 \vdash u_2 \vdash \ldots$ is fully in the co-basis U\B; thus, P(u) holds. To show (b), for an arbitrary v, prove the contraposition $\neg P(v) \rightarrow \neg\forall u(u \dashv v \rightarrow P(u))$. But $\neg P(v)$ means that $v = u_n$, for some $n \in N$, and $\neg\forall u(u \dashv v \rightarrow P(u))$ means that there exists an $m \in N$, such that $u_m \dashv u_n$ and that $\neg P(u_m)$. Such an $m \in N$ really exists – namely, this is n+1. To show (c), notice that $\neg P(u_n)$ holds for any $n \in N$. QED.

4 Successors and limits

The transfinite induction condition has two "sub-conditions", called "successor case condition" and "limit case condition", which are important in proofs by induction and definitions by induction. Unlike transfinite induction, the reduction principle has only the reduction condition $\forall x(\forall y((y \dashv x) \rightarrow P(x)) \rightarrow P(x)))$. Our next task is to represent the reduction principle in the same form as the transfinite induction, by generalizing the notions "successor" and "limit" from ordinals to arbitrary objects of a universe.

For a universe U, and u, $v \in U$, call v "successor" of u (or u "predecessor" of v), and denote this as "$x \cdot \dashv y$", if there does not exist a $w \in U$, such that $u \dashv w$ and $w \dashv v$. Call an element v of U a *successor element* (or *predecessor element*) of U and denote this as "successor(v)", if there exists an element u of U, such that v is successor of u. Call x a limit element in U and denote this as "limit(x)", if x is not a successor element in U. Dually, one can define what is a "predecessor in U" and a "lower limit in U", but we will employ here only the "successor" and "limit" terms, where the term "limit" is supposed to be "upper limit". Notice that a successor (predecessor) in U can have many predecessors (successors), but this is not important for our purposes. These terms are needed to enable presenting the reduction principle in another form by replacing the "induction step condition" in the transfinite induction principle with the conjunction of following formulas:

$$S \stackrel{\text{def}}{=} \forall x(\text{successor}(x) \rightarrow \forall u((u \cdot \dashv x) \,\&\, P(u) \rightarrow P(x))),$$
$$L \stackrel{\text{def}}{=} \forall x(\text{limit}(x) \rightarrow ((\forall u(u \cdot \dashv x) \rightarrow P(u)) \rightarrow P(x))).$$

Say the new form of reduction principle to be in form "form 2". Thus, the reduction principle in form 2 looks like this:

$$(\forall x(B(x) \to P(x)) \ \& \ S \ \& \ L \to \forall x P(x).$$

Proposition. The induction principle in main form and the induction principle in form 2 are equivalent.

To prove this proposition, it is enough to prove that the formula $\forall x(\forall u((u \dashv x) \to P(u)) \to P(x))$ is equivalent to S & L.

5 An algebraic set theory based on the ideas of universics

A finite set $A = \{a_1, \ldots, a_n\}$ can be treated as built in a process which starts from the empty set and continues by applying an operation of "bringing" the objects $a_1, \ldots, a_n$ one by one into the set until the set is ready. Only when an object is "brought into" A, it can be said to be an "element" of A – untill an object is brought into A, it remains what it was – an "object", which for the universe of a set theory is either an atom or a set. The operation of "bringing in" an object x into a set y is said in this paper to be "adduction" (which in Latin means exactly "bring in") and is denoted as ":", or, if one wants to have a symbol for the inverse operation, then the symbol "$\cdot:$" with its symmetric "$:\cdot$" symbol can be used (as it was recommended in Part 1 in the context of algebraization of universics). The symbol ":" comes from the notation of a class as $\{x : P(x)\}$ (where it has a slightly different meaning, since in it "x" is a set, but "P(x)" is a property), and ":" as a symbol of operation plays same fundamental role in our algebraic set theory as the symbol "$\in$" of a relation plays in set theory.

Thus, the adduction of x into y is denoted as "$x : y$" and it can be treated as the set $\{x\} \cup y$. Notice, that the following equivalence holds:

$$x : y = y \text{ iff } x \in y.$$

This equivalence can be used for the algebraization of set theory (see "Method 1" in Part 1.)

There are two synonyms, "adjunction" of "adduction", used in lit-

erature, and "adjunction" is more frequently used than "adduction". But there are many reasons why the term "adduction" should be preferred in universics: (a) this word is from the same family as the term "reduction" ("induction" and "deduction"), (b) This Latin word has the intended meaning "to bring in" in Latin, (c) the word "adjunction" sounds close to "adjointness" and the search engines wrongly deliver links related with category theory when the search is made by the key word "adjunction".

The idea about adduction goes back to Zermelo who encountered it in connection with a form of induction principle applied in the universe of set theory [2], and in definition of natural numbers by the induction using the "diagonal form" operation $\{x\} \cup x$. But Alfred Tarski seems to have been the first to have paid really close attention to this operation [3] and to have used a special form of induction principle based on this operation, which served for axiomatizing the theory of finite sets. In [4], this operation was used to develop a theory of inheritably finite sets and a theory of recursive functions over them.

The theory of sets developed here is actually an algebraic theory of baseless universes regarded as "magmas" – collections, equipped with a binary operation denoted here as ":" – algebraic counterparts of universes, which are universal algebras and which can be called "algebraic universes". The objects populating such an algebraic universe should be treated as "set-like" objects, but of a kind different from "collections" (i.e. sets and classes), in particular, due to having repeating elements. These objects are called here "aggregations" and their theory is called "theory of aggregations" or "aggregation theory" (similar to "theory of sets" but "set theory") and is denoted as **A**.

An algebraic universe is a structure of the type (U, ":"), which corresponds to the structure (U, "$\in$") in the sense that "$\in$" is defined through ":" like this: $(x \in y) \stackrel{\text{def}}{=} (x : y = y)$. The theory **A** is considered as an extension of the theory **W**, where the reduction symbol "$\dashv$" needs to be replaced by the symbol "$\in$". Notice, that reduction principle, as well as any other formulas using the symbol "$\in$", can be represented in the language of the theory **A**, because the symbol "$\in$" of relation is definable in **A** through the symbol of operation ":".

In addition to reduction principle which provides for well-founded universes, the theory **A** has a number of algebraic axioms (the names of axioms are indicated in parentheses):

$$(x : x) : z = x : z, \text{ (left idempotency)}$$
$$(x : y) : z = (y : x) : z, \text{ (left commutativity)}$$
$$x : (y : z) = y : z \text{ iff } x = y \vee x \in z. \text{ (adduction axiom)}$$

6 The intuition behind aggregations

Notice that if the set $\{s_1, s_2, \ldots, s_n\}$ is denoted as S, which is usually represented as $S = \{s_1, s_2, \ldots, s_n\}$, then this fact can be represented as $s_1 : (s_2 : \ldots (s_n : S)) = S$. Also, due to left idempotency and left commutativity axioms, if the right associativity rule for denotation of expressions using ":" is adopted, then the parentheses can be dropped and the colon ":" can be replaced by comma, this manner:

$s_1, s_2, \ldots, s_n : S$. This reminds the class notation $\{x : P(x)\}$, except that instead of one "generic element" x, a list of elements $s_1, s_2, \ldots, s_n$ is indicated, the outer curly braces are not used, and instead of a property P(x), a "set-like" object called "aggregation" is used.

The representation of a finite aggregation as the list of its elements and as an identifier of it shows that an aggregation can be treated as a set represented by this list together with its identifier, and this is indicative of the intuition of "multi-set". Once many identifiers can be used for the same set, the aggregations can be treated as "*multi-identity* objects" or "*multi-id* objects".

Another intuitive interpretation of aggregations is by treating them as common names (versus proper names), and the relation "$x \in y$" as "x is named as y" or "y is name of x". With this interpretation in mind a name of exactly one named object is a proper name, and an aggregation of this kind represents a singleton of set theory.

In such a set theory, each urelement can be treated as a multi-id empty set, and there is no need to "singularize" the empty set as a special object. If though one wants such an object to be singularized, then a constant "0" can be added to the language of theory **A**, and an axiom like this $\forall x\ (\neg\ x : 0)$ to the theory **A**. In a theory with

this constant a unary operation called "individuation" or ("singleton formation" or "singularization") can be defined like this: $\{x\} \stackrel{\text{def}}{=} x : 0$. One can say that the aggregation theory with constant "0" also contains "proper names" formed by this operation.

7 Conclusion

The results of this papers sound to the author as meeting the Harvey Friedman's expectations for a foundation for mathematics, expressed in his "foundational issue" stated in one of his messages in the automated email list FOM ("Foundations of Mathematics" https://cs.nyu.edu/pipermail/fom/1997-November/000143.html), where the relevant paragraph is this:

FOUNDATIONAL ISSUE. Is there an alternative adequate foundation for mathematics that is based on "inductive construction?" In particular, one wants to capture set theory viewed as an inductive construction. If not, one wants to construct a significant portion of set theory as an inductive construction.

The aggregations introduced in this paper have the features of multi-sets and are more specific to informatics with its "data structures", where an object can have many copies as opposed to mathematics with its "abstract structures" where an abstract object occurs in only one copy. The extensionality axiom is specific to the abstractions called sets and to mathematics presented in terms of such abstractions, but it is utterly non-specific to aggregations, which can have potentially an infinite number of copies, and to informatics operating with data structures which can be presented in terms of aggregations. Thus, the belief of the author is that the universics, in general, and the aggregation theory, in particular, can serve as a foundational theory also for the informatics.

References

[1] N. Bourbaki. *Elements of mathematics.* Commutative algebra, Addison-Wesley, 1972.

[2] E. Zermelo. *Sur les ensembles finis et le principle de l'induction complète,* Acta Mathematica, vol. 32 (1909), pp. 185–93.

[3] A.Tarski, S.Givant. *A formalization of set theory without variables.* Colloquium Publications, vol. 41, American Mathematical Society, Providence, R.I., 1987.

[4] L. Kirby. *Finitary Set theory. Notre Dame Journal of Formal Logic,* Volume 50, Number 3, 2009, pp. 227-244.

[5] P. Bernays. *A System of Axiomatic Set Theory. Part I,* The Journal of Symbolic Logic (Association for Symbolic Logic) **2** (1), 1937, pp. 65–77. (adjunction axiom)

Ioachim Drugus

Received August 2, 2015

Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
5 Academiei str., Chisinau, MD-2028, Moldova
E–mail: ioachim.drugus@math.md

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Extensionality, Proper Classes, and Quantum Non-Individuality

William J. Greenberg

Abstract

Here I broach three questions: (1) What is a class? (2) What constitutes a class as a set? and (3) What are the logical parameters of non-individuality? I answer (1) by proposing two axioms, which describe classes in such a way as to make clear what safe and problematic classes have in common.

I answer (2) by extensionalizing some safe ones – empty, sum, union and power classes – whose individuation by their members constitutes these as sets. Problematic ones – classes of non-self-membered sets, classes of sets, classes of self-identicals, and so on – are not extensionalized, for their constitution as sets leads to paradox.

I answer (3) by exhibiting a parallelism between proper classes and quantum particles, the former unindividuated by their members and the latter unindividuated by their properties. On my analysis, proper classes and quantum particles – unlike sets and classical particles, whose members and properties individuate them – each lack individuality. For both, as it turns out, lack self-identity.

1 Three Principles

Modulo the deductive apparatus of First-Order Logic with Weak Identity[1], Russell's Paradox follows from three principles: *Unrestricted Extensionality*, *Restricted Comprehension*, and *Unrestricted Pairing*. Concerning the first of these Michael Potter writes:

[1] In FOL=$_W$: $x = y \to y = x$ and $(x = y \;\&\; y = z) \to x = z$ are theses but $x = x$ is not. Every proof in FOL=$_W$ is a proof in FOL=. So FOL=$_W$ is a sub-theory of FOL=.

> Various theories of [classes] have been proposed since the 1900s. What they all share is the axiom of extensionality, which asserts that if x and y are [classes] then
>
> $$\forall z(z \in x \leftrightarrow z \in y) \rightarrow x = y.$$
>
> The fact that they share this is just a matter of definition: objects which do not satisfy extensionality are not [classes]. ([12])

Restricted Comprehension says that for every condition $P(x)$, some y contains just the sets satisfying $P(x)$.

$$\exists y \forall x(x \in y \leftrightarrow (\text{set } x \ \& \ Px)).$$

And *Unrestricted Pairing* says that for every w, u and some y: identity-with-w or identity-with-u is necessary and sufficient for membership in y:

$$\forall x(x \in y \leftrightarrow (x = w \vee x = u)).$$

Individually, each of these is plausible. But no consistent theory features all three. For (A,B,C) prove (D),[2] engendering Russell's Paradox.

(A)	$\forall x \forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow x = y)$	(*Unrestricted Extensionality*)
(B)	$\exists y \forall x(x \in y \leftrightarrow \text{set } \& \ Px)$	(*Restricted Comprehension*)
(C)	$\forall w \forall u \exists y \forall x(x \in y \leftrightarrow x = w \vee x = u)$	(*Unrestricted Pairing*)
(D)	$\exists y \forall x(x \in y \leftrightarrow Px)$	(*Unrestricted Comprehension*)

[2]

1. $\forall z(z \in x \leftrightarrow z \in y) \rightarrow x = y$	*Unrestricted Extensionality*
2. $\exists y \forall x(x \in y \leftrightarrow (\text{set } x \ \& \ Px))$	*Restricted Comprehension*
3. $\forall t \forall w \exists y \forall x(x \in y \leftrightarrow (x = t \vee x = w))$	*Unrestricted Pairing*
4. $\forall t \forall w \forall x \exists y(x \in y \leftrightarrow x = t \vee x = w)$	3, Quantifier Shift
5. $\forall x \exists y(x \in y \leftrightarrow x = x)$	4,UI
6. $\forall x \exists y(x = x \rightarrow x \in y)$	5
7. $\forall x[x = x \rightarrow \exists y(x \in y)]$	6
8. $\forall x(x = x) \rightarrow \forall x \exists y(x \in y)$	7
9. $\forall x(x = x)$	Corollary of A
10. $\forall x \exists y(x \in y)$	8,9
11. $\forall x(\text{set } x)$	10, definition of *set*
12. $\exists y \forall x(x \in y \leftrightarrow Px)$	2,11

(D) can be avoided by replacing (B) with (B′), as in Zermelo Set Theory;

$$(\mathrm{B}')\ \forall z \exists y \forall x (x \in y \leftrightarrow (x \in z\ \&\ Px)) \quad (\textit{Separation})$$

or by replacing (C) with (C') as in NBG*, a sub-theory of NBG;

$$\begin{array}{rl} (\mathrm{C}')\ \forall w \forall u((\text{set } w\ \&\ \text{set } u) \rightarrow & (\textit{Restricted} \\ \exists y \forall x (x \in y \leftrightarrow (x = w \vee x = u))) & \textit{Pairing}) \end{array}$$

or by replacing (A) with (A′), as in NBG$^-$: an NBG-like theory with *Restricted Extensionality* and *Unrestricted Pairing*:

$$\begin{array}{lll} (\mathrm{A}') & \forall x \forall y (\forall z (z \in x \leftrightarrow z \in y) \rightarrow & (\textit{Restricted} \\ & ((\text{set } x\ \&\ \text{set } y) \leftrightarrow x = y)) & \textit{Extensionality}) \\ (\mathrm{B}) & \exists y \forall x (x \in y \leftrightarrow (\text{set } x\ \&\ Px)) & (\textit{Restricted} \\ & & \textit{Comprehension}) \\ (\mathrm{C}) & \forall w \forall u \exists y \forall x (x \in y \leftrightarrow x = w \vee x = u) & (\textit{Unrestricted} \\ & & \textit{Pairing}) \end{array}$$

2 Proper Classes in NBG$^-$

From (A′) it follows that identity is reflexive for sets (T1) but irreflexive for proper classes (T2).

T1: $\forall x (x = x \leftrightarrow \text{set } x)$

T2: $\forall x (\neg(x = x) \leftrightarrow \text{prop } x)$[3]

From (B) it follows that there is a class of non-self-membered sets (T3),

T3: $\exists y \forall x (x \in y \leftrightarrow (\text{set } x\ \&\ \neg(x \in x)))$,

which is not a set but a proper class (T4)–and thus not self-identical (T5).

T4: $\forall y (\forall x (x \in y \leftrightarrow (\text{set } x\ \&\ \neg(x \in x))) \rightarrow \text{prop } y)$

T5: $\forall y (\forall x (x \in y \leftrightarrow (\text{set } x\ \&\ \neg(x \in x))) \rightarrow \neg(y = y))$

[3] Prop $x \stackrel{\text{def}}{=} \neg(\text{set } x)$

Hence there is no universe class (T6).

T6: $\neg\exists y\forall x(x \in y)$

(A′, B) secure an empty class (T7); a pair class (T8: *aka* C); a sum class (T9); a power class (T10); a class of self-identicals (T11); and a class of sets (T12).

T7: $\exists y\forall x\neg(x \in y)$ (*Empty Class*)[4]

T8: $\forall w\forall u\exists y\forall x(x \in y \leftrightarrow x = w \vee x = u)$ (*Pair Class*)[5]

T9: $\forall z\exists y\forall x(x \in y \leftrightarrow \exists w(w \in z \;\&\; x \in w))$ (*Sum Class*)[6]

T10: $\forall z\exists y\forall x(x \in y \leftrightarrow (\text{set } x \;\&\; \forall w(w \in x \rightarrow w \in z)))$ (*Power Class*)

T11: $\exists y\forall x(x \in y \leftrightarrow x = x)$ (*Class of Self-Identicals*)[7]

T12: $\exists y\forall x(x \in y \leftrightarrow \text{set } x)$ (*Class of Sets*)

Remark 1: "Set x" doesn't appear on the right-hand side of T8 or T9 because it is redundant.

[4]
1. $\forall x(x \in y \leftrightarrow (\text{set } x \;\&\; \neg(x = x)))$ B, EI
2. $\exists x(x \in y) \leftrightarrow \exists x(\text{set } x \;\&\; \neg(x = x))$ 1
3. $\forall x(\text{set } x \leftrightarrow x = x)$ A'
4. $\neg\exists x(x \in y)$. 2, 3

[5]
1. Show $\forall a\forall b\exists y\forall x(x \in y \leftrightarrow x = a \vee x = b)$
2. Show $\exists y\forall x(x \in y \leftrightarrow x = a \vee x = b)$
3. $\forall a\forall b\exists y\forall x(x \in y \leftrightarrow (\text{set } x \;\&\; (x = a \vee x = b)))$ B
4. $x \in y \leftrightarrow (\text{set } x \;\&\; (x = a \vee x = b))$ 3, UI, EI
5. $(x = a \vee x = b) \rightarrow x = x$ "=" is weakly reflexive
6. $x = x \rightarrow \text{set } x$ A'
7. $(x = a \vee x = b) \rightarrow \text{set } x$ 5, 6
8. $(x = a \vee x = b) \rightarrow x \in y$ 4,6,7
9. $x \in y \rightarrow (x = a \vee x = b)$ 4
10. $x \in y \leftrightarrow (x = a \vee x = b)$ 8, 9
11. $\forall x(x \in y \leftrightarrow x = a \vee x = b)$ 10, UG
12. $\exists y\forall x(x \in y \leftrightarrow x = a \vee x = b)$ 11, EG: Cancel *Show* at line 2
13. $\forall a\forall b\exists y\forall x(x \in y \leftrightarrow x = a \vee x = b)$ 2, UG: Cancel *Show* at line 1

[6]
1. Show $\forall z\exists y\forall x(x \in y \leftrightarrow \exists w(w \in z \;\&\; x \in w))$
2. $\forall z\exists y\forall x(x \in y \leftrightarrow (\text{set } x \;\&\; \exists w(w \in z \;\&\; x \in w)))$
3. $\exists w(w \in z \;\&\; x \in w) \rightarrow \text{set } x$
4. $\forall z\exists y\forall x(x \in y \leftrightarrow \exists w(w \in z \;\&\; x \in w))$ Cancel *Show* at line 1

[7]
1. $\exists y\forall x(x \in y \leftrightarrow (\text{set } x \;\&\; x = x))$ B
2. $\text{set } x \leftrightarrow x = x$ A'
3. $\exists y\forall x(x \in y \leftrightarrow x = x)$ 1, 2
4. $\exists y\forall x(x \in y \leftrightarrow \text{set } x)$ 2,3

Remark 2: From T8 it follows that the "singleton" of a non-self-identical is empty.[8]

3 Some Classes Are Not Sets

Conventional wisdom decrees that some classes are not sets, either because they are infinite totalities "too large" to be sets ([8], 44 *ff*; [11], 264 *ff*), or because their members "are not all present at any rank of the iterative hierarchy". ([7], 104) According to John Bell, however, infinite totalities are not problematic per se. He writes:

> ... set theory... as originally formulated, does contain contradictions, which result not from admitting infinite totalities per se, but rather from countenancing totalities consisting of all entities of a certain abstract kind, "manys" which, on pain of contradiction, cannot be regarded as "ones". So it was in truth not the finite/infinite opposition, but rather the one/many opposition, which led set theory to inconsistency. This is well illustrated by the infamous Russell paradox, discovered in 1901. ([1], 173)

My treatment of Russell's paradox squares with Bell's observation. *Restricted Comprehension* provides for classes of non-self-membered sets, but on pain of contradiction these "manys" cannot be treated as "one", as would be the case if they were subject to *Unrestricted Extensionality*. Indeed, from *Restricted Comprehension* it follows that every predicate is associated with (perhaps empty) classes of sets which satisfy it. In NBG$^-$ (and its extensions), whether such "manys" can be "ones" – that is, sets – depends not on their size or rank, but on whether their "oneness" would spawn contradiction ([1], op. cit.).

[8] "Consider a thing, a say, and its unit set $\{a\}$. . . If anything x is not a member of the unit set $\{a\}$ then that thing x is not a. And conversely, if anything x is not a then that thing x is not a member of the unit set $\{a\}$." ([3], 82)

4 Proper Classes, Sets, and Models

Suppose $\forall z(z \in x \leftrightarrow z \in y)$. Are x and y identical? Are x and y sets? Unlike NBG*, in NBG$^-$ identity and set-hood go hand-in-hand: equi-membered x and y are identical *iff* these are sets.[9] But from (A′,B) it does not follow that equi-membered classes are sets. Therefore, although (A′,B) prove T7-T12, they do not make equi-membered classes identical: *unlike sets, classes are not individuated by their members.*

(A′,B) are satisfied only in domains which include a non-self-identical, non-element. But such domains violate model-theoretic restrictions on individuals enunciated by Ruth Marcus, who in "Dispensing With Possibilia" writes:

> The notion of an individual object or thing is an indispensable primitive for theories of meaning grounded in standard model theoretic semantics. One begins with a domain of individuals, and there are no prima facie constraints as to what counts as an individual except those of a most general and seemingly redundant kind. *Each individual must be distinct from every other and identical to itself* (emphasis added). ([9], 39)

5 NBG$^-$ and NBG*

NBG$_-$ and NBG* are *deviations*[10] of one another, for $\neg\forall x(x = x)$ is a theorem of NBG$^-$ and $\forall x(x = x)$ a theorem of NBG*. I will now show that NBG* and NBG$^-$ are *definitional extensions* of one another as well. To show this I will define "=" in terms of "I" and "$\in$" in NBG*,

[9] In *Elementary Logic*, Mates writes, "...we have explicated the term 'relation' in such a way that whatever cannot be a member of a set cannot be related by any relation. Thus insofar as identity is a relation in this sense, such a thing cannot even stand in this relation to itself. This would hold not only of the set of all objects that are not members of themselves, but also of sets described by phrases that give no hint of impending difficulties. The problem is closely related to *Russell's Antinomy*, and once again every way out seems unintuitive." ([10], 157-8)

[10] "One system is a deviation of another if it shares the vocabulary of the first, but has a different system of theorems/valid inferences." ([5], 3)

and "I" in terms of "$\in$" in NBG$^-$; and then show that NBG* $\vdash$ NBG$^-$, and NBG$^-$ $\vdash$ NBG*.

NBG*:

($\in$)	1*:	$\forall x\forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow \forall z(x \in z \leftrightarrow y \in z))$
(I)	2*:	$\forall x\forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow x\mathrm{I}y)$
	3*:	$\forall x\forall y(x\mathrm{I}y \rightarrow \forall z(z \in x \leftrightarrow z \in y))$
(Set)	4*:	$\exists y\forall x(x \in y \leftrightarrow (\text{set } x \ \&\ Px))$
	5*:	$\forall w\forall u((\text{set } w \ \&\ \text{set } u) \rightarrow \exists y\forall x(x \in y \leftrightarrow (x\mathrm{I}w \vee x\mathrm{I}u)))$
(Def)	D1*:	$\text{set } x \stackrel{\text{def}}{=} \exists y(x \in y)$
	D2*:	$x = y \stackrel{\text{def}}{=} (x\mathrm{I}y \ \&\ \text{set } x \ \&\ \text{set } y)$

NBG$^-$:

($\in$)	1$^-$:	$\forall x\forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow \forall z(x \in z \leftrightarrow y \in z))$
(=)	2$^-$:	$\forall x\forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow ((\text{set } x \ \&\ \text{set } y) \leftrightarrow x = y))$
	3$^-$:	$\forall x\forall y(x = y \rightarrow \forall z(z \in x \leftrightarrow z \in y))$
(Set)	4$^-$:	$\exists y\forall x(x \in y \leftrightarrow (\text{set } x \ \&\ Px))$
(Def)	D1$^-$:	$\text{set } x \stackrel{\text{def}}{=} \exists y(x \in y)$
	D2$^-$:	$x\mathrm{I}y \stackrel{\text{def}}{=} \forall z(z \in x \leftrightarrow z \in y)$

NBG* $\vdash$ NBG$^-$: Since $1^- = 1^*$ and $4^- = 4^*$, to show that NBG* $\vdash$ NBG$^-$ I will show that NBG* $\vdash 2^-, 3^-$

Proof of 2$^-$:

1. Show $\forall x\forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow ((\text{set } x \ \&\ \text{set } y) \leftrightarrow x = y))$	
2. $\forall z(z \in x \leftrightarrow z \in y)$	Assume
3. Show $(\text{set } x \ \&\ \text{set } y) \leftrightarrow x = y$	
4. $x = y \stackrel{\text{def}}{=} (x\mathrm{I}y \ \&\ \text{set } x \ \&\ \text{set } y)$	D2*
5. $x = y \rightarrow (\text{set } \&\ \text{set } y)$	4
6. Show $(\text{set } x \ \&\ \text{set } y) \rightarrow x = y$	
7. $\text{set } x \ \&\ \text{set } y$	Assume
8. Show $x = y$	
9. $x\mathrm{I}y \ \&\ \text{set } x \ \&\ \text{set } y$	2*, 2, 7
10. $x = y$	9, D2*: Cancel *Show* line 8
11. $(\text{set } x \ \&\ \text{set } y) \rightarrow x = y$	7, 8: Cancel *Show* line 6
12. $(\text{set } x \ \&\ \text{set } y) \leftrightarrow x = y$	5, 6: Cancel *Show* line 3
13. $\forall x\forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow ((\text{set } x \ \&\ \text{set } y) \leftrightarrow x = y))$	2, 3: Cancel *Show* line 1

Proof of 3^- :

1. Show $\forall x\forall y(x = y \rightarrow \forall z(z \in x \leftrightarrow z \in y))$	
2. $x = y$	Assume
3. Show $\forall z(z \in x \leftrightarrow z \in y))$	
4. $x = y \stackrel{\text{def}}{=} (x\text{I}y$ & set x & set $y)$	D2*
5. $x\text{I}y$	2, 4
6. $\forall x\forall y(x\text{I}y \rightarrow \forall z(z \in x \leftrightarrow z \in y))$	3*
7. $\forall z(z \in x \leftrightarrow z \in y)$	5, 6: Cancel *Show* line 3
8. $\forall x\forall y(x\text{I}y \rightarrow \forall z(z \in x \leftrightarrow z \in y))$	2, 3: Cancel *Show* line 1

NBG$^-$ $\vdash$ NBG*: Since $1^- = 1^*$ and $4^- = 4^*$, to show that NBG$^-$ $\vdash$ NBG* I will show that NBG$^-$ $\vdash$ 2*, 3*,5*.

Proof of 2*:

1. Show $\forall x\forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow x\text{I}y)$	
2. $\forall z(z \in x \leftrightarrow z \in y)$	Assume
3. Show $x\text{I}y$	
4. $x\text{I}y \stackrel{\text{def}}{=} \forall\text{z}(\text{z} \in \text{x} \leftrightarrow \text{z} \in \text{y})$	D2$^-$
5. $x\text{I}y$	2, 4: Cancel *Show* line 3
6. $\forall x\forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow x\text{I}y)$	2, 3: Cancel *Show* line 1

Proof of 3*:

1. Show $\forall x\forall y(x\text{I}y \rightarrow \forall\text{z}(\text{z} \in \text{x} \leftrightarrow \text{z} \in \text{y}))$	
2. $x\text{I}y \stackrel{\text{def}}{=} \forall\text{z}(\text{z} \in \text{x} \leftrightarrow \text{z} \in \text{y})$	D2$^-$
3. $\forall x\forall y(x\text{I}y \rightarrow \forall\text{z}(\text{z} \in \text{x} \leftrightarrow \text{z} \in \text{y}))$	2: Cancel *Show* line 1

Proof of 5*:

1. Show $\forall w\forall u((\text{set } w \text{ \& set } u) \rightarrow \exists y\forall x(x \in y \leftrightarrow (x\text{I}w \vee \text{x}u)))$	
2. set w & set u	Assume
3. Show $\exists y\forall x(x \in y \leftrightarrow (x\text{I}w \vee \text{x}u))$	
4. $\exists y\forall x(x \in y \leftrightarrow (\text{set } x \text{ \& } Px))$	4^-
5. $\exists y\forall x(x \in y \leftrightarrow (\text{set } x \text{ \& } x = w \vee x = u))$	Instance of 4
6. $(x = w \vee x = u) \rightarrow \text{set } x$	1^-
7. $x \in y \rightarrow \text{set } x$	$D2^-$
8. $\exists y\forall x(x \in y \leftrightarrow (x = w \vee x = u))$	5, 6, 7
9. $x = w \leftrightarrow \forall z(z \in x \leftrightarrow z \in w)$	3^-
10. $x = u \leftrightarrow \forall z(z \in x \leftrightarrow z \in u)$	3^-
11. $\exists y\forall x(x \in y \leftrightarrow (\forall z(z \in x \leftrightarrow z \in w) \vee \forall z(z \in x \leftrightarrow z \in)))$	8, 9, 10
12. $\exists y\forall x(x \in y \leftrightarrow x\text{I}w \vee \text{x}u)$	11, D2$^-$: Cancel *Show* line 3
13. $\forall w\forall u((\text{set } w \text{ \& set } u) \rightarrow \exists y\forall x(x \in y \leftrightarrow (x\text{I}w \vee \text{x}u)))$	2, 3: Cancel *Show* line 1

Each a definitional extension of the other, NBG* and NBG$^-$are accordingly equi-consistent. The question thus arises, Which of these two systems–NBG* (in which identity is reflexive and proper classes are individuated by their members), or NBG$^-$ (in which identity is non-reflexive and proper classes are not individuated by their members)–should be employed as a setting for theories in which all sets are classes, but some classes are not sets?

6 Classes Into Sets

(5^-, 6^-, 7^-, 8^-) constitute as sets: pair classes, sum classes, power classes, and sub-classes of sets. For it follows from (5^-,6^-, 7^-, 8^-) that these are individuated by their members.

5^-: $\forall y(\forall x(x \in y \leftrightarrow (x = a \vee x = b)) \rightarrow \text{set } y)$ (*Pair Set*)
6^-: $\forall z\forall y(\forall x(x \in y \leftrightarrow \exists w(w \in z \,\&\, x \in w)) \rightarrow \text{set } y)$ (*Sum Set*)
7^-: $\forall z\forall y(\forall x(x \in y \leftrightarrow (\text{set } x \,\&\, \forall w(w \in x \rightarrow w \in z))) \rightarrow \text{set } y)$ (*Power Set*)
8^-: $\forall z\forall y(\forall x(x \in z \rightarrow x \in y) \rightarrow (\text{set } y \rightarrow \text{set } z))$ (*Subsets*)

To guarantee an empty set, Z and its extensions require an axiom of infinity or an axiom of set existence; and Lemmon's NBG requires an axiom, "set $\emptyset$". ([6], 46) In NBG$^-$ no such apparatus is required, for (1^-, 2^-, 5^-) guarantee an empty set.

Thus (1^-, 2^-, 5^-) prove T13,

T13: $\forall y(\forall x\neg(x \in y) \rightarrow \text{set } y)$ (*Empty Set*)

which together with T7: $\exists y\forall x\neg(x \in y)$ establish a unique empty set.

Proof of T13:

Suppose y empty. From (T7, T8, 5^-) we have $\exists z(\text{set } z \,\&\, \forall x(x \in z \leftrightarrow x = y))$, and so by EI: set z & $\forall x(x \in z \leftrightarrow x = y)$. Hence $\exists x(x \in z) \leftrightarrow \exists x(x = y)$. Now suppose, contrary to T13, that y is a proper class. Then $\neg(y = y)$, $\neg\exists x(x = y)$, and $\neg\exists x(x \in z)$, so that z and y have the same members. So because z is a set, from T14 it follows that y is a set:

T14: $\forall x \forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow (\text{set } x \leftrightarrow \text{set } y))$ (*EquiEqui*)[11]

So if y is a proper class, y is a set. Hence y is a set.

7 NBG$^-$ and Foundation

4^- provides for a class of self-membered sets:

T15: $\exists y \forall x(x \in y \leftrightarrow x \in x)$ (*Class of self-membered sets*)

The *Anti-Foundation* axiom 9^- would constitute this class as a set.

9^-: $\forall y(\forall x(x \in y \leftrightarrow x \in x) \rightarrow \text{set } y)$ (*Anti-Foundation*)

But a *Foundation* axiom such as $9^{-\prime}$ would constitute such a class as a proper class.

$9^{-\prime}$:$\forall y(\forall x(x \in y \leftrightarrow x \in x) \rightarrow \neg(\text{set } y))$ (*Foundation*)

8 NBG$^-$ and the Identity of Indiscernibles

A set-theoretic analogue of Leibniz's principle of the Identity of Indiscernibles is:

$\forall x \forall y(\forall z(x \in z \leftrightarrow y \in z) \rightarrow x = y)$ (*Unrestricted PII*)

Unrestricted *PII* is refuted in NBG$^-$. For by satisfying $\neg(x = x)$, proper classes refute $\forall x(x = x)$, a corollary of *PII*. *PII* must thus be restricted, by excluding proper classes from its range of application, *thus*:

$\forall x \forall y(\forall z(x \in z \leftrightarrow y \in z) \rightarrow ((\text{set } x \ \& \ \text{set } y) \leftrightarrow x = y))$ (*Restricted PII*)

And to save pairing and extensionality, which together prove unrestricted *PII*, either pairing or extensionality must be restricted, as in NBG* or NBG$^-$.

[11] If x,y are equi-membered, from 1^- it follows that x is an element iff y is an element. So set x iff set y.

9 Quasi-Set Theories, Non-Individuality, and $\neg(\mathbf{x = x})$

Quasi-Set theories deal with collections of indistinguishable objects such as quantum particles. Such theories recognize two kinds of entities: *M-Atoms*, which "have the properties of standard *Ur-elemente* of ZFU"; and *m-atoms*, which "represent the elementary basic entities of quantum physics". To m-atoms "the concept of identity does not apply." In Quasi-Set theories, "this exclusion is achieved by restricting the concept of formula: expressions like $x = y$ are not well formed if x and y denote m-atoms. The equality symbol is not a primitive logical symbol...". ([4], p 276] Whereof they cannot speak, thereof French and Krause must remain silent. Because identity does not figure among the primitive notions of Quasi-Set theory, they are unable to ask whether their *non-self-identity* grounds the non-individuality of quantum particles.

To be an individual is to have a property that exactly one thing has. Call this an *individuating* property. For x *not* to have an individuating property, it must be the case that for every P, if x has P there is some y such that y has P and $\neg(x = y)$. But quantum particles share *all* their properties. Hence if x is a quantum particle, there is no P such that x and only x has P.

Nevertheless, it is a theorem of Zermelo-Fraenkel set theory with *ur-elemente* (ZFU) that everything in its domain belongs to a unit set – a set of which it is the sole member. Extrapolating from sets to properties, everything in the domain of ZFU thus has an individuating property. Quantum particles however *lack* individuating properties. Quantum particles thus *falsify* the pairing axiom of ZFU, according to which *everything* belongs to a unit set – unless one gerrymanders identity by making it inapplicable to quantum particles.[12]

[12]Dean Rickles writes, "An immediate problem with the denial of primitive identities is, then, that it is unclear how one is able to support set theory. . . (I owe this point to Steven French). There are ways of accommodating the denial of primitive identities through the use of 'quasi-set theory' in which the identity relation is not a well-formed formula for indistinguishable objects (see French & Krause [1999] and

In contrast, the logical setting I propose for non-individuality is one in which identity *is* applicable to quantum particles. Indeed, it is this applicability which expands logic's domain to include non-self-identicals, whose breach of the Identity of Indiscernibles establishes their non-individuality.

10 Summary and Conclusion

A: $\forall x \forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow x = y)$ (*Unrestricted Extensionality*)
B: $\exists y \forall x(x \in y \leftrightarrow \text{set } x \ \& \ Px)$ (*Restricted Comprehension*)
C: $\forall w \forall u \exists y \forall x(x \in y \leftrightarrow x = w \vee x = u)$ (*Unrestricted Pairing*)

Modulo a background logic in which identity is a partial equivalence relation, the inconsistency of (A,B,C) can be resolved by replacing B with B′, as in Z*;

Z*	
A: $\forall x \forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow x = y)$	*Unrestricted Extensionality*
B′: $\forall z \exists y \forall x(x \in y \leftrightarrow (x \in z \ \& Px))$	*Separation*
C: $\forall w \forall u \exists y \forall x(x \in y \leftrightarrow x = w \quad \vee x = u)$	*Unrestricted Pairing*

or by replacing C with C', as in NBG*;

NBG*	
A: $\forall x \forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow x = y)$	*Unrestricted Extensionality*
B: $\exists y \forall x(x \in y \leftrightarrow (\text{set } x \ \& \ Px))$	*Restricted Comprehension*
C′: $\forall w \forall u((\text{set } w \ \& \ \text{set } u) \rightarrow \exists y \forall x(x \in y \leftrightarrow (x = w \vee x = u)))$	*Restricted Pairing*

or by replacing A with A', as in NBG$^-$.

Z* and NBG* are sub-theories of Z and NBG. NBG$^-$ and NBG* are deviations *and* definitional extensions of one another.

Highlighting the rivalry of NBG* and NBG$^-$, I have proposed NBG$^-$ – in which identity is reflexive for sets and classical particles, but

Krause [1992])". ([13], 106)

$\mathbf{NBG^-}$	
A′: $\forall x \forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow ((\text{set } x \ \& \ \text{set } y) \leftrightarrow x = y))$	*Restricted Extensionality*
B: $\exists y \forall x(x \in y \leftrightarrow (\exists z(x \in z) \ \& \ Px))$	*Restricted Comprehension*
C: $\forall w \forall u \exists y \forall x(x \in y \leftrightarrow x = w \vee x = u)$	*Unrestricted Pairing*

irreflexive for proper classes and quantum particles – as a setting for class and set theory and framework for quantum non-individuality.[13]

References

[1] Bell, John L. Oppositions and paradoxes in mathematics and philosophy. *Axiomathes* 15.2:165-80, 2005.

[2] Ben-Menahem, Yemima, editor. *Hilary Putnam*. Cambridge University Press, Cambridge, UK, 2005.

[3] Bigelow, John. Sets are haecceities. In D. M. Armstrong et al, editors, *Ontology, causality and mind*, pages 73-96. Cambridge University Press, Cambridge, UK, 1993.

[13] As things now stand, non-self-identicals in NBG are non-elements, making their identification with quantum particles problematic. To surmount this obstacle, a more restrictive definition of "set" is required:

set $x \stackrel{\text{def}}{=} \exists y(x \in y \ \& \ \forall z(z \in y \rightarrow z = x))$ ($\stackrel{\text{def}}{=}$ *Set*)

From ($\stackrel{\text{def}}{=}$ *Set*) and "set $x \leftrightarrow x = x$" (a corollary of 2 – 2 holding under both definitions of "set"), it follows that possession of an individuating property is necessary and sufficient for self-identity:

$x = x \leftrightarrow \exists y(x \in y \ \& \ \forall z(z \in y \rightarrow z = x))$ (*Ind*)

Hence $\neg(x = x)$ if, and only if, x is not a member of any unit class.

$\neg(x = x) \leftrightarrow \forall y(\neg(x \in y) \vee \exists z(z \in y \ \& \ \neg(z = x)))$ (*Ind*)

This will be the case if x is not a member of *any* class (think "proper" classes); or if every class that x belongs to is multi-membered (think quantum particles). In both cases, x can be said to *lack individuality*.

[4] French, Steven, and Décio Krause. *Identity in physics*: *A historical, philosophical, and formal analysis*. Clarendon Press, Oxford, UK, 2006.

[5] Haack, Susan. *Deviant logic*: *Some philosophical issues*. Cambridge University Press Archive, Cambridge, UK, 1974.

[6] Lemmon, Edward John. *Introduction to axiomatic set theory*. Routledge & K. Paul, London/New York, 1969.

[7] Lewis, David K. *On the plurality of worlds*. (Vol. 322). Blackwell, Oxford, UK, 1986.

[8] Maddy, Penelope. *Naturalism in mathematics*. Clarendon Press, Oxford, UK, 1997.

[9] Marcus, Ruth Barcan. Dispensing with possibilia. *Proceedings and Addresses of the American Philosophical Association*. American Philosophical Association, 1975.

[10] Mates, Benson. *Elementary logic*. 2^{nd} edition. Oxford University Press, New York, 1972.

[11] Moore, Gregory H. *Zermelo's axiom of choice: Its origins, development, and influence*. Springer-Verlag, New York/Heidelberg/Berlin, 1982.

[12] Potter, Michael. *Different systems of set theory*. Sourced from http://www.scribd.com/doc/172940806/Different-Systems-of-Set-Theory.

[13] Rickles, Dean. *Symmetry, structure, and space time*. Elsevier, Amsterdam, the Netherlands, 2008.

William J. Greenberg

USA
E–mail: wgreenb@gmail.com

Received July 10, 2015

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Fundamental theorems of extensional untyped λ-calculus revisited

Alexandre Lyaletsky

Abstract

This paper presents new proofs of three following fundamental theorems of the untyped extensional λ-calculus: the η-Postponement theorem, the $\beta\eta$-Normal form theorem, and the Normalization theorem for $\beta\eta$-reduction. These proofs do not involve any special extensions of the standard language of λ-terms but nevertheless are shorter and much more comprehensive than their known analogues.

Keywords: extensional untyped λ-calculus, $\beta\eta$-reduction, postponement of η-reduction, η-Postponement theorem, $\beta\eta$-Normal form theorem, Normalization theorem for $\beta\eta$-reduction.

1 Introduction

The untyped version of the λ-calculus is considered.

Its variables are denoted by symbols x, y and z, λ-terms by t, p, q, u and w, redeces by $\triangle$ (of cause, indices are sometimes used). All the other denotations used in the paper are completely standard or otherwise will be introduced separately.

Throughout the paper the variable convention is assumed to be satisfied; hence the conditions $x \notin t$ and $x \notin FV(t)$ say the same.

Recall some basic facts concerning η- and $\beta\eta$-reduction. By definition, the notion η of η-reduction is $\{ < \lambda x.wx, w > \mid x \notin w \}$, the notion $\beta\eta$ of $\beta\eta$-reduction is $\beta \cup \eta$. The notions of η- and $\beta\eta$-reduction induce, in the usual way, the dictionaries of their derivative notions (such as η- and $\beta\eta$-redeces, the relations of one-step and multi-step η-

and $\beta\eta$-reduction, η- and $\beta\eta$-reduction sequences, etc.). Remark that the notion of η-reduction is strongly normalizing since the contraction of an η-redex in any λ-term decreases its length.

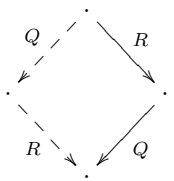
The extensional untyped λ-calculus studies properties of the notion of $\beta\eta$-reduction as well as of its derivative relations, especially $\twoheadrightarrow_{\beta\eta}$ (multi-step $\beta\eta$-reduction) and $=_{\beta\eta}$ ($\beta\eta$-convertibility). Along with the Church-Rosser theorem for $\beta\eta$-reduction (and not taking the results on λ-representability into account), the most important results in the extensional λ-calculus are: 1) the η-Postponement theorem, 2) the $\beta\eta$-Normal form theorem, 3) the Normalization theorem for $\beta\eta$-reduction.

In the paper, new proofs of the theorems 1) – 3) are constructed. These proofs do not involve any special extensions of the standard language of λ-terms but nevertheless are shorter and much more comprehensive than their original or known analogues. (For example, the original proof of the theorem 3) by J.W.Klop takes over 20 pages and is technically very complicated.) The new proofs are arranged in the following logical order: 1) $\Rightarrow$ 2) $\Rightarrow$ 3).

2 Postponable binary relations

Our proof of the η-Postponement theorem exploits some general properties which are more convenient to be observed and studied in the general set-theoretic situation.

Definition 1. Given a set A and binary relations Q and R on A, then R is said to be *postponable* after Q if the following diagram holds:



(Here and in the sequel, the language of diagrams of binary relations is used. In the general case, such a diagram is a configuration on the

plane consisting of points some of which may be labelled by elements of a fixed set A, and arrows between points, each obligatorily labelled by a binary relation on A. Each arrow can be of two sorts: usual or dotted. If a diagram contains a usual arrow from a to b and labelled by R, then the latter expresses that $a\,R\,b$; if a point has no label, then it is considered to be bounded by a universal quantifier (restricted by A). At that, precisely those arrows are dotted that lead to or start with the elements, the existence of which is being claimed (together with the conditions imposed by the labels). Thus, each diagram (containing at least one dotted arrow) determines a certain implicative statement. For example, the diagram from the last definition means the following:

$$\forall a, b, c \in A \left[\, a\,R\,b \;\;\&\;\; b\,Q\,c \;\;\Rightarrow\;\; \exists\, d \in A\,(a\,Q\,d \;\;\&\;\; d\,R\,c) \,\right],$$

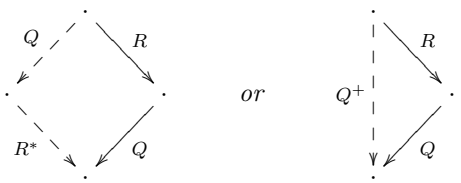
i.e. that $R \circ Q \subseteq Q \circ R$, where $\circ$ denotes the usual composition of binary relations.)

Note that if binary relations $Q = f$ and $R = g$ are functions, then g is postponable after f if and only if $g \circ f = f \circ g$, that is f and g commute with each other. Therefore, in this case f is postponable after g as well. However in the general case the latter is not valid, i.e. the notion of postponability is not symmetric.

By R^+ and R^* denote, resp., the transitive and reflexive-transitive closures of a binary relation R.

It can easily be proved that if R is postponable after Q, then R^* is postponable after Q^*. Containing this statement as a particular case, the following result can be viewed as its natural generalization.

Postponement Lemma. *Let Q and R be binary relations on a set A such that for any triple of elements of A, at least one of the following diagrams holds:*



Then R^* is postponable after Q^* and $(Q \cup R)^* = Q^* \circ R^*$.

Proof. Since $R^* \circ Q^* \subseteq (Q \cup R)^*$, for proving the postponability, it is sufficient to show that $(Q \cup R)^* \subseteq Q^* \circ R^*$, which also substantiates the second statement (the inclusion opposite to the latter holds trivially). Let a and c be any elements of A with $a\,(Q \cup R)^* c$. Obviously, it can be assumed that $a \neq c$. Then there is, for some elements $b_1, \ldots, b_{n-1} \in A$, a valid sequence of the form

$$b_0\, S_0\, b_1\, S_1\, b_2\, S_2 \ldots S_{n-2}\, b_{n-1}\, S_{n-1}\, b_n\,, \tag{1}$$

where $b_0 = a$, $b_n = c$, and $S_i \in \{Q, R\}$ for every $i \in \{0, n-1\}$.

Consider its leftmost "two-step" segment of the form $b_k R\, b_{k+1} Q\, b_{k+2}$ (if there is no such a segment, there is nothing to prove). If, for simplicity, the "Q-prefix" of (1) is empty, then (1) has the following form:

$$b_0 R\, b_1 R\, b_2 R \ldots R\, b_{k-1} R\, \underline{b_k\, R\, b_{k+1} Q\, b_{k+2}}\, S_{k+2}\, b_{k+3}\, S_{k+3} \ldots S_{n-1} b_n\,. \tag{2}$$

Applying one of the diagrams from the conditions of the lemma to the underlined segment, one of the following sequences will be obtained:

$$b_0 R \ldots R\, b_{k-1} R\, \underline{b_k\, Q\, b'_{k+1} R\, b'_{k+1,\,1} R \ldots R\, b'_{k+1,\,m} R\, b_{k+2}}\, S_{k+2} \ldots S_{n-1}\, b_n$$

(in the case of the left diagram), where m is a natural number, or

$$b_0 R \ldots R\, b_{k-1} R\, \underline{b_k\, Q\, b'_{k+1,\,1} Q \ldots Q\, b'_{k+1,\,m} Q\, b_{k+2}}\, S_{k+2} \ldots S_{n-1}\, b_n$$

(in the case of the right diagram), where m is a positive natural number.

Comparing the obtained sequences with the previous one, notice that in the both cases, the position of the leftmost occurrence of Q is one item to the left than that was in (2). Therefore, the proof can be completed by induction, applied to the set of sequences of the form (1) that is considered to be lexicographically ordered in accordance with the positions of all occurrences of Q in a sequence under consideration when reading it from left to right. $\square$

Remark. The Postponement lemma states less than that was proved. Actually, the above given proof determines an algorithm of reconstructing each sequence (1) to a form $b_0\, Q \ldots Q\, d\, R \ldots R\, b_n$ (of cause, provided the diagrams are satisfied). This algorithm will be referred to as postponing R after Q.

3 Postponement of η-reduction

Note that when considering diagrams over the set of λ-terms some arrows of which are labelled by one-step reductions, it is often convenient to introduce additional labels for (some of) these arrows for indicating the contracted redex occurrences. (Examples are given below.)

Given a one-step $\beta\eta$-reduction sequence $\sigma\colon t_1 \overset{\triangle}{\longrightarrow}_{\beta\eta} t_2$. If $\triangle_1$ is such a $\beta\eta$-redex occurrence in t_1 that has exactly one residual in t_2 (w.r.t. σ), then the latter will be denoted by $\overrightarrow{\triangle}_1$. If $\triangle_2$ is a $\beta\eta$-redex occurrence in t_2, then it can be trivially verified that there can be at most one $\beta\eta$-redex occurrence in t_1 for which $\triangle$ is a residual of (or belongs to the set of residuals of); in this case, it is denoted by $\overleftarrow{\triangle}_2$, i.e. $\overleftarrow{\triangle}_2$ is a "coresidual" of $\triangle_2$ w.r.t. σ.

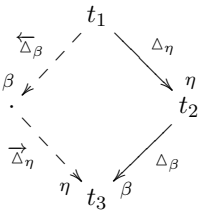
η-Postponement Theorem. [1; 2] *Every finite $\beta\eta$-reduction sequence $\sigma\colon t_1 \twoheadrightarrow_{\beta\eta} t_2$ can be reconstructed into a sequence of the form $\sigma'\colon t_1 \twoheadrightarrow_{\beta} t' \twoheadrightarrow_{\eta} t_2$, for some λ-term t'.*

Proof. Let us verify the diagrams from the conditions of the Postponement lemma (with $Q = \longrightarrow_\beta$ and $R = \longrightarrow_\eta$). For a given two-step $\beta\eta$-reduction sequence of the form $t_1 \overset{\triangle_\eta}{\longrightarrow}_\eta t_2 \overset{\triangle_\beta}{\longrightarrow}_\beta t_3$, note that the coresidual $\overleftarrow{\triangle}_\beta$ always exists and is always a β-redex occurrence in t_1. Let $\triangle_\eta \equiv \lambda x.wx$, $\overleftarrow{\triangle}_\beta \equiv (\lambda z.p)q$ and $\triangle_\beta \equiv (\lambda y.p')q'$. Consider all possible cases of the mutual locations of the redeces $\triangle_\eta$ and $\overleftarrow{\triangle}_\beta$ in t_1:

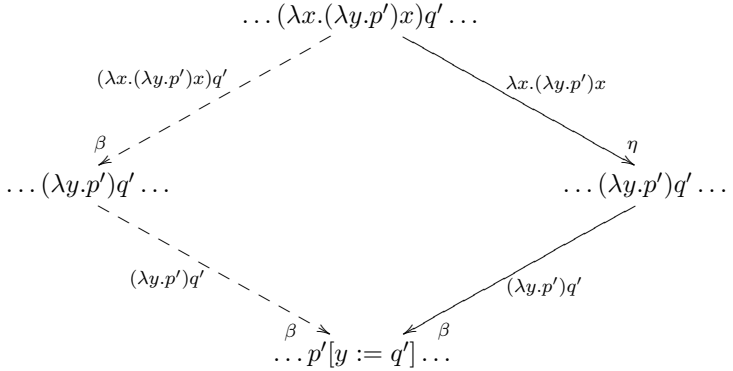
1. $\triangle_\eta \cap \overleftarrow{\triangle}_\beta = \varnothing$		
2. $\triangle_\eta \supset \overleftarrow{\triangle}_\beta$		
3. $\triangle_\eta \subset \overleftarrow{\triangle}_\beta$	3.1. $\triangle_\eta \equiv \lambda x.(\lambda y.p')x$	(hence $z \equiv x$ and $p \equiv (\lambda y.p')x$)
	3.2. $\triangle_\eta \subseteq p$	(hence $z \equiv y$ and $q \equiv q'$)
	3.3. $\triangle_\eta \equiv \lambda x.q'x$	(hence $z \equiv y$ and $p \equiv p'$)
	3.4. $\triangle_\eta \subset q$	(hence $z \equiv y$ and $p \equiv p'$)

The cases 1, 2 and 3.2 are trivial: one only needs to reverse the order of contractions (first, to contract $\overleftarrow{\triangle}_\beta$ in t_1 and then, to contract the residual $\overrightarrow{\triangle}_\eta$ of $\triangle_\eta$ in the resulting term), which all lead to a diagram

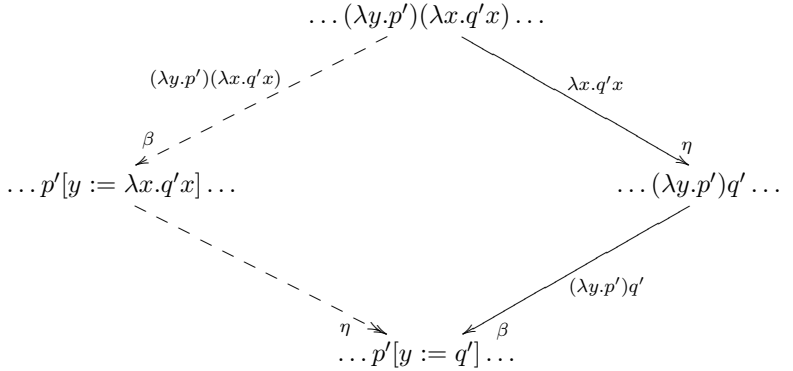
of the form:



Case 3.1:



Case 3.3:



Finally, the verification of the case 3.4 leads to a diagram similar to that obtained when considering the case 3.3. $\square$

4 $\beta\eta$-Normal forms

Note that for a λ-term of the form $(\lambda x.wx)q$, where $x \notin w$, the both contractions $(\lambda x.wx)q \xrightarrow{(\lambda x.wx)q}_{\beta} wq$ and $(\lambda x.wx)q \xrightarrow{\lambda x.wx}_{\eta} wq$ lead to the same result wq.

Definition 2. An η-redex occurrence $\triangle_\eta \equiv \lambda x.wx$ in a λ-term t is called *β-replaceable*, if $\triangle_\eta$ is a re-part of some β-redex occurrence in t, i.e. there is a term q such that $(\lambda x.wx)q$ is a β-redex occurrence in t.

Lemma 1. *Given a finite η-reduction sequence $\sigma\colon t \twoheadrightarrow_\eta t'$ in which neither of the contracted η-redeces is β-replaceable. If t' is a β-normal form, then so is t.*

Proof. Obviously, it is sufficient to prove the lemma for the case of a one-step η-reduction sequence $\sigma\colon t \xrightarrow{\triangle_\eta}_\eta t'$. If t is not a β-normal form, then it contains some β-redex occurrence $(\lambda x.p)q$ and hence $t \equiv \ldots(\lambda x.p)q\ldots$. Since $\triangle_\eta$ is not β-replaceable, it follows that $\triangle_\eta \not\equiv \lambda x.p$. Then, evidently, $(\lambda x.p)q$ has a nonempty residual in t' which is a β-redex occurrence. Thus, t' is not a β-normal form. $\square$

$\beta\eta$-Normal Form Theorem. [1; 3] *An arbitrary λ-term t has a $\beta\eta$-normal form $\Leftrightarrow$ t has a β-normal form.*

Proof. The sufficiency is obvious, since if t has a β-normal form t', then any η-redex contraction in t' does not create new β-redeces and decreases the length of t'.

Let us prove the necessity. Suppose a λ-term t has a $\beta\eta$-normal form t'. By the Church-Rosser theorem for $\beta\eta$-reduction ([1; 2]), then there is a $\beta\eta$-reduction sequence $\sigma\colon t \twoheadrightarrow_{\beta\eta} t'$. Moreover, it can be assumed without loss of generality that neither of the η-redeces being contracted in σ is β-replaceable (since any such an η-redex can be replaced in σ with the corresponding β-redex). Furthermore, from the analysis of the diagrams from the proof of the η-Postponement theorem it can be concluded that when postponing η-reduction in σ, no contracted β-replaceable η-redeces can emerge. Therefore, there exists a $\beta\eta$-reduction sequence of the form $\sigma'\colon t \twoheadrightarrow_\beta u \twoheadrightarrow_\eta t'$, for some λ-term u,

that does not contract neither of the β-replaceable η-redeces. Lemma 1 finally implies that u is the needed β-normal form of t. $\square$

5 Normalization theorem for $\beta\eta$-reduction

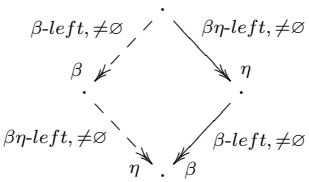
Recall that for a given notion R of reduction and λ-term t, an R-leftmost redex is any such an R-redex occurrence in t, the position (in t) of the first symbol of which cannot be strictly to the right of the position of the first symbol of any other R-redex occurrence. Therefore, this notion is not deterministic in the general case, i.e. for some notion R of reduction, a term t may have two or more distinct leftmost R-redex occurrences. By this reason, the latter notion is sometimes strengthened to the notion of the R-leftmost-outermost redex occurrence which is always unique in every λ-term (if any).

As to the $\beta\eta$-reduction, the notion of a $\beta\eta$-leftmost redex occurrence is not deterministic as well. However this is a small problem, since the only possibility for the ambiguity in this case is when considering terms with subterm occurrences of the form $(\lambda x.px)q$, where $x \notin p$ (having two distinct leftmost $\beta\eta$-redex occurrences: $\lambda x.px$ and $(\lambda x.px)q$, the both contractions of which lead to the same result pq).

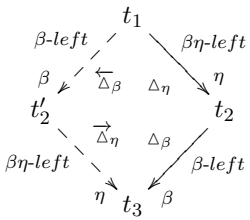
By $left_{\beta\eta}$ denote the so-called $\beta\eta$-leftmost strategy which always contracts, in every λ-term t, the $\beta\eta$-leftmost-outermost redex occurrence (if any). It generates, for each term t, a certain finite or infinite $\beta\eta$-reduction sequence starting with t which is also called $\beta\eta$-leftmost. Analogously, by $left_{\beta}$ denote the β-leftmost strategy.

We write $t \xrightarrow[\beta\eta\text{-}left]{} t'$ and $t \xrightarrow[\beta\text{-}left]{} t'$ instead of, resp., $t' = left_{\beta\eta}(t)$ and $t' = left_{\beta}(t)$. The notation $t \xrightarrow[\beta\eta\text{-}left]{\triangle}_{\eta} t'$ means that $t \xrightarrow[\beta\eta\text{-}left]{} t'$ and $t \xrightarrow{\triangle}_{\eta} t'$. Therefore, $\xrightarrow[\beta\eta\text{-}left]{}_{\eta}$ can be considered as a binary relation on the set of λ-terms; by $\xrightarrow[\beta\eta\text{-}left,\neq\varnothing]{}_{\eta}$ denote its transitive closure. The relations $\xrightarrow[\beta\eta\text{-}left]{}_{\beta}$ and $\xrightarrow[\beta\eta\text{-}left,\neq\varnothing]{}_{\beta}$ are introduced analogously, the same concerns $\xrightarrow[\beta\text{-}left]{}_{\beta}$ and $\xrightarrow[\beta\text{-}left,\neq\varnothing]{}_{\beta}$.

Lemma 2. *The following diagram holds:*

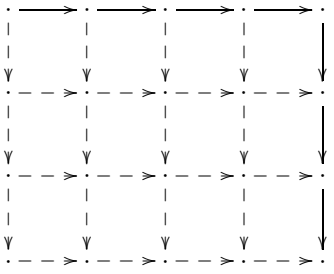


Proof. First consider the case of a two-step reduction sequence of the form $t_1 \xrightarrow[\beta\eta\text{-}left]{\triangle_\eta} \eta\, t_2 \xrightarrow[\beta\text{-}left]{\triangle_\beta} \beta\, t_3$. Since $\triangle_\eta$ is the $\beta\eta$-leftmost-outermost redex, it follows that $\triangle_\eta$ is strictly to the left of $\overleftarrow{\triangle}_\beta$ in t_1. Obviously, this is possible only in the cases 1 and 2 from the proof of the η-Postponement theorem, which, as noted there, leads to the following diagram:



(at that, evidently, $\overleftarrow{\triangle}_\beta$ is indeed the β-leftmost redex occurrence in t_1 and $\overrightarrow{\triangle}_\eta$ the $\beta\eta$-leftmost-outermost occurrence in t'_2).

Now the general case can be concluded from the following typical example of a diagram can arise under such conditions (in which all the labels are omitted due to the triviality of its construction):



□

Recall that a $\beta\eta$-strategy f is called normalizing if whenever a term t has a $\beta\eta$-normal form, $f^n(t)$ is a $\beta\eta$-normal form for some natural number n.

Normalization Theorem for $\beta\eta$-Reduction. [4] *The strategy $left_{\beta\eta}$ is normalizing.*

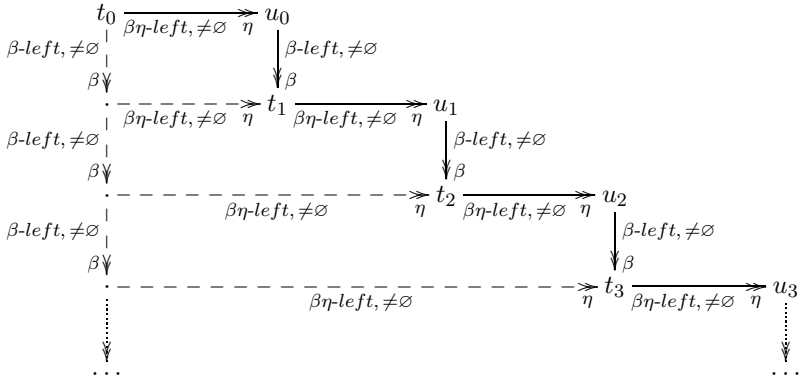
Proof. Supposing the contrary, there is a term t having a $\beta\eta$-normal form, the $\beta\eta$-leftmost reduction sequence σ of which is infinite. It will be proved, by means of postponing η-reduction with the help of Lemma 2, that σ can be reconstructed into the infinite β-leftmost reduction sequence (starting with t), which leads to a contradiction: indeed, by the Normalization theorem for β-reduction ([1; 3]), then t does not have a β-normal form and by the $\beta\eta$-Normal form theorem, t does not have a $\beta\eta$-normal form as well.

It can be assumed without loss of generality that σ contracts at least one η-redex (otherwise, σ is already β-leftmost) and, moreover, that it contracts the infinite number of η-redeces (otherwise, exclude from σ such an initial segment that the result contracts β-redeces only, which sends back to the previous case). On the other hand, σ should contract the infinite number of β-redeces, since the notion of η-reduction is strongly normalizing. In addition to all these conditions, it can also be assumed, for definiteness, that σ starts with an η-redex contraction (otherwise, exclude from σ its β-prefix). Then σ can be represented in a form of the following infinite sequence:

$$\sigma\colon t_0 \xrightarrow[\beta\eta\text{-}left,\neq\varnothing]{}\!\!\!\!\!\!\rightarrow_\eta u_0 \xrightarrow[\beta\eta\text{-}left,\neq\varnothing]{}\!\!\!\!\!\!\rightarrow_\beta t_1 \xrightarrow[\beta\eta\text{-}left,\neq\varnothing]{}\!\!\!\!\!\!\rightarrow_\eta u_1 \xrightarrow[\beta\eta\text{-}left,\neq\varnothing]{}\!\!\!\!\!\!\rightarrow_\beta \dots,$$

where $t_0 \equiv t$ (i.e. σ is being divided into alternating β- and η-segments).

Finally, notice that every $\beta\eta$-leftmost reduction sequence that contracts β-redeces only is evidently the β-leftmost reduction sequence as well. Now the following infinite diagram can be constructed with the help of Lemma 2:



Its vertical line determines the β-leftmost reduction sequence starting with $t_0 \equiv t$. Thus, the latter is indeed infinite, just as expected. $\square$

An interested reader is invited to compare the constructed proofs with their original or known analogues. The detailed references to the latter are contained in the following table compiled for his convenience:

η-Postponement theorem	[1, pp. 384 – 386]; [2, pp. 132 – 135]
$\beta\eta$-Normal form theorem	[1, pp. 384 – 386]; [3, pp. 313 – 314]
Normalization theorem for $\beta\eta$-reduction	[4, pp. 279 – 290]

References

[1] H. Barendregt, *The Lambda Calculus. Its Syntax and Semantics* (revised ed.), Elsevier (1984).

[2] H. Curry, R. Feys, and W. Craig, *Combinatory Logic* (vol. I), North-Holland: Amsterdam (1958).

[3] H. Curry, J. Hindley, and J. Seldin, *Combinatory Logic* (vol. II), North-Holland: Amsterdam (1972).

[4] J.W. Klop, *Combinatory reduction systems* (PhD thesis), Utrecht university (1980).

Alexandre Lyaletsky Received July 22, 2015

Alexandre Lyaletsky
Institution: Kiev National Taras Shevchenko university (faculty of cybernetics)
Phone: +38 067 4086768
E–mail: foraal@mail.ru

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Semantic Properties of Logics of Quasiary Predicates

Mykola Nikitchenko, Stepan Shkilniak

Abstract

In the paper we investigate semantic properties of program-oriented algebras and logics defined for classes of quasiary predicates. Informally speaking, such predicates are partial predicates defined over partial states (partial assignments) of variables. Conventional n-ary predicates can be considered as a special case of quasiary predicates. We define first-order algebras and logics of quasiary non-deterministic predicates and investigate their semantic properties. Obtained results can be used to prove logic validity and completeness.

Keywords: First-order logic, quasiary predicate, partial predicate, non-deterministic predicate.

1 Introduction

Mathematical logic is one of the basic disciplines for computer science. To use effectively mathematical logic it is important to construct logical systems that are adequate for problems considered in computer science. Classical logic, despite its numerous advantages, has some restrictions for its use in this area. For example, classical logic is based on the class of total n-ary predicates, while in computer science partial and non-deterministic predicates often appear. Therefore many logical systems which better reflect properties of such kind were constructed [1, 2]. One of specific features for computer science is quasiarity of predicates. Such predicates are partial predicates defined over partial states (partial assignments) of variables. Conventional n-ary predicates

can be considered as a special case of quasiary predicates. In our previous works [3, 4, 5] we investigated the class of partial deterministic (single-valued) predicates and constructed corresponding logics.

This paper aims to develop a semantic basis for construction of logics of non-deterministic (many-valued) quasiary predicates. To realize this idea we should first construct predicate algebras using *composition-nominative approach* [6]. Terms of such algebras specify the language of logic. Then we should define interpretation mappings. At last, we should construct calculi of sequent type for defined logics.

It is important to admit that constructed logics better reflect specifics of computer science problems, but the opposite side of this feature is that the methods of logic investigation turn out to be more complicated. In this paper we concentrate on semantic properties of first-order logics of non-deterministic quasiary predicates.

The rest of the paper is structured as follows. In Section 2 we define first-order algebras of quasiary predicates. In Sections 3 we define logics of quasiary predicates. Section 4 is devoted to semantic properties of such logics. In Section 5 the main consequence relations are specified and their properties are studied. In Section 6 conclusions are formulated.

2 First-order algebras of quasiary predicates

Let V be a nonempty *set of names*. According to tradition, names from V are also called *variables*. Let A be a set of basic values ($A \neq \emptyset$). Given V and A, the class ${}^{V}A$ of nominative sets is defined as the class of all partial mappings from V to A, thus, ${}^{V}A = V \xrightarrow{p} A$. Informally speaking, nominative sets represent states of variables.

Though nominative sets are defined as mappings, we follow mathematical traditions and also use set-like notation for these objects. In particular, the notation $d = [v_i \mapsto a_i \mid i \in I]$ describes a nominative set d; the notation $v_i \mapsto a_i \in_n d$ means that $d(v_i)$ is defined and its value is a_i ($d(v_i) \downarrow = a_i$). The main operation for nominative sets is a total unary parametric *renomination* $r^{v_1,\ldots,v_n}_{x_1,\ldots,x_n} : {}^{V}A \xrightarrow{t} {}^{V}A$ where

$v_1, ..., v_n, x_1, ..., x_n \in V$, $v_1, ..., v_n$ are distinct names, $n \geq 0$, which is defined by the following formula:

$$r^{v_1,...,v_n}_{x_1,...,x_n}(d) =$$
$$= [v \mapsto a \in_n d \mid v \notin \{v_1, ..., v_n\}] \cup [v_i \mapsto d(x_i) \mid d(x_i)\downarrow, i \in \{1, ..., n\}].$$

Intuitively, given d this operation yields a new nominative set changing the values of $v_1, ..., v_n$ to the values of $x_1, ..., x_n$ respectively. We also use simpler notation for this formula: $r^{\bar{v}}_{\bar{x}}(d) = d\nabla\bar{v} \mapsto d(\bar{x})$. Also note that we treat a parameter $^{v_1,...,v_n}_{x_1,...,x_n}$ as a total mapping from $\{v_1, ..., v_n\}$ into $\{x_1, ..., x_n\}$ thus parameters obtained by pairs permutations are identical.

Operation of deleting a component with a name v from a nominative set d is denoted $d|_{-v}$. Notation $d =_{-v} d'$ means that $d_{-v} = d'_{-v}$. The set of *assigned names* (*variables*) in d is defined by the formula

$$asn(d) = \{v \in V \mid v \mapsto a \in_n d \text{ for some } a \in A\}.$$

Let $Bool = \{F, T\}$ be a set of Boolean values.

Let $PrR^V_A = {}^VA \xrightarrow{r} Bool$ be the set of all non-deterministic (relational) predicates over ^VA. Such predicates are called *non-deterministic (relational) quasiary predicates*. The term 'relational' means that graphs of such predicates are binary relations from $^VA \times Bool$. Note that non-determinism in logic was intensively studied, see, for example, [7].

We will also use set-theoretic notations for quasiary predicates.

Full image of $d \in {}^VA$ under $p \in PrR^V_A$ is defined by the formula $p([d]) = \{b \in Bool \mid (d, b) \in p\}$.

For $p \in PrR^V_A$ the truth and falsity domains of p are respectively

$$T(p) = \{d \in {}^VA \mid (d, T) \in p\} \text{ and } F(p) = \{d \in {}^VA \mid (d, F) \in p\}.$$

Arrows $\xrightarrow{t}$ and $\xrightarrow{p}$ specify total and partial operations respectively.

Notations not defined in this paper are understood in a sense of [4].

Considering predicates from PrR^V_A in set-theoretic style we can speak about such operations as union $\cup$ and intersection $\cap$. The following statement is obvious.

Lemma 1. *The set* $< PrR^V_A; \cup, \cap >$ *is a complete distributive lattice.*

The greatest and the least elements of this lattice are denoted $\top_A^V$ and $\bot_A^V$ respectively. For these elements $T(\top_A^V) = {}^VA$, $F(\top_A^V) = {}^VA$, $T(\bot_A^V) = \emptyset$, $F(\bot_A^V) = \emptyset$.

Operations over PrR_A^V are called *compositions*. The set $C(V)$ of first-order compositions is $\{\vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x\}$. Compositions have the following types:

$\vee : PrR_A^V \times PrR_A^V \xrightarrow{t} PrR_A^V$; $\neg, R_{x_1,\ldots,x_n}^{v_1,\ldots,v_n}, \exists x : PrR_A^V \xrightarrow{t} PrR_A^V$ and are defined by the following formulas ($p, q \in PrR_A^V$):

- $T(p \vee q) = T(p) \cup T(q)$; $F(p \vee q) = F(p) \cap F(q)$;
- $T(\neg p) = F(p)$; $F(\neg p) = T(p)$;
- $T(R_{\bar{x}}^{\bar{v}}(p)) = \{d \in {}^VA \mid r_{\bar{x}}^{\bar{v}}(d) \in T(p)\}$;
  $F(R_{\bar{x}}^{\bar{v}}(p)) = \{d \in {}^VA \mid r_{\bar{x}}^{\bar{v}}(d) \in F(p)\}$;
- $T(\exists xp) = \{d \in {}^VA \mid d\nabla x \mapsto a \in T(p) \text{ for some } a \in A\}$;
  $F(\exists xp) = \{d \in {}^VA \mid d\nabla x \mapsto a \in F(p) \text{ for all } a \in A\}$.

Here $d\nabla x \mapsto a = [v \mapsto c \in_n d \mid v \neq x] \cup [x \mapsto a]$. Conventional notation is $d[v \mapsto a]$.

Please note that definitions of compositions are similar to strong Kleene's connectives and quantifiers.

Also note that parametric compositions of existential quantification and renomination can also represent classes of compositions. Thus, notation $\exists x$ can represent one composition, when x is fixed, or a class $\{\exists x \mid x \in V\}$ of such compositions for various names.

A pair $AQR(V, A) =< PrR_A^V; C(V) >$ is called *a first-order algebra of non-deterministic quasiary predicates*.

It is not difficult to prove the following statement.

Lemma 2. *Singleton sets* $\{\top_A^V\}$ *and* $\{\bot_A^V\}$ *are sub-algebras of algebra* $AQR(V, A)$.

Algebras $AQR(V, A)$ (for various A) form a semantic base for the constructed first-order pure quasiary predicate logic L^{QR} (called also quasiary logic). Let us now proceed with formal definitions.

3 First-order pure quasiary logic

To define a logic we should define its semantic component, syntactic component, and interpretational component [3, 4, 5]. Semantics of the logic under consideration is specified by algebras of the type $AQR(V, A)$ (for various A), so, we proceed with syntactic component of the logic.

3.1 Syntactic component

A syntactic component specifies the language of L^{QR}. Let $Cs(V)$ be *a set of composition symbols* that represent compositions in algebras defined above – $Cs(V) = \{\vee, \neg, R^{\bar{v}}_{\bar{x}}, \exists x\}$. For simplicity, we use the same notation for symbols of compositions and compositions themselves.

Let Ps be a set of *predicate symbols*. A triple $\Sigma^Q = (V, Cs(V), Ps)$ is a language *signature*. Given Σ^Q , we inductively define the language of L^{QR} – *the set of formulas* $Fr(\Sigma^Q)$:

1) if $P \in Ps$, then $P \in Fr(\Sigma^Q)$; such formulas are called atomic;

2) if Φ, $\Psi \in Fr(\Sigma^Q)$, then $(\Phi \vee \Psi) \in Fr(\Sigma^Q)$;

3) if $\Phi \in Fr(\Sigma^Q)$, then $(\neg\Phi) \in Fr(\Sigma^Q)$;

4) if $\Phi \in Fr(\Sigma^Q)$, $v_1, ..., v_n$, $x_1, ..., x_n \in V$, $v_1, ..., v_n$ are distinct names, $n \geq 0$, then $(R^{v_1,...,v_n}_{x_1,...,x_n}(\Phi)) \in Fr(\Sigma^Q)$;
for such formulas notation $R^{v_1,...,v_n}_{x_1,...,x_n}\Phi$ or $R^{\bar{v}}_{\bar{x}}\Phi$ can be also used;

5) if $\Phi \in Fr(\Sigma^Q)$, $x \in V$, then $(\exists x\Phi) \in Fr(\Sigma^Q)$.

Extra brackets can be omitted using conventional rules of operation priorities.

3.2 Interpretational component

Given Σ^Q and nonempty set A we can define an algebra of quasiary predicates $AQR(V, A) =< PrR^V_A; C(V) >$. Composition symbols have fixed interpretation, but we additionally need interpretation I^{Ps} :

$Ps \xrightarrow{t} PrR_A^V$ of predicate symbols; obtained predicates are called *basic predicates*. A tuple $J = (\Sigma^Q, A, I^{Ps})$ is called *an interpretation*.

Formulas and interpretations in L^{QR} are called L^{QR}*-formulas* and L^{QR}*-interpretations* respectively. Usually the prefix L^{QR} is omitted. Given a formula Φ and an interpretation J we can speak of an *interpretation of* Φ in J. It is denoted by Φ_J.

3.3 Extensions of L^{QR}

The logic L^{QR} being a rather powerful logic still is not expressive enough to represent transformations required for proving its completeness. Therefore we introduce its two extensions: L^{UR} — *a logic with unessential variables*, and L_ε^{UR} — a logic with unessential variables and a parametric total deterministic *variable unassignment predicate* εz which checks if a variable z is unassigned in a given nominative set.

To define L^{UR} we should specify its semantic, syntactic, and interpretational components.

Let U be an infinite set of variables such that $V \cap U = \emptyset$. Variables from U are called *unessential variables* (analogs of fresh variables in classical logic) that should not affect the formula meanings.

Algebras

$$AQR(V \cup U, A) =< Pr_A^{V \cup U}; C(V \cup U) >$$

(for different A) form a semantic base for L^{UR}.

A syntactic component is specified by the set of formulas $Fr(\Sigma^U)$ where $\Sigma^U = (V \cup U, Cs(V \cup U), Ps)$ is the *signature* of L^{UR}.

An interpretational component restricts the class of L^{UR}-interpretations in such a way that interpretations of predicate symbols are neither sensitive to the values of the component with an unessential variable u in nominative sets, nor to presence of such components. Formally, a variable $u \in U$ is *unessential in an interpretation of predicate symbols* I^{Ps} if $I^{Ps}(P)([d]) = I^{Ps}(P)([d'])$ for all $P \in Ps$, $d, d' \in {}^{V \cup U}A$ such that $d =_{-u} d'$.

The following statement is obvious.

Lemma 3. *L^{UR} is a model-theoretic conservative extension of L^{QR}.*

Note that given $p \in PrR_A^{V\cup U}$ and $v \in V \cup U$ we say that v is unessential for p if $p([d]) = p([d'])$ for any $d, d' \in {}^{V\cup U}A$ such that $d =_{-u} d'$.

The next logic L_ε^{UR} is an extension of L^{UR} by a null-ary parametric composition (predicate) εz ($z \in V \cup U$) defined in interpretation J by the following formulas:

$$T(\varepsilon z_J) = \{d \in {}^V A \mid z \notin asn(d)\},$$
$$F(\varepsilon z_J) = \{d \in {}^V A \mid z \in asn(d)\}.$$

Thus, for this logic the set of compositions is equal to $\{\vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, \varepsilon z\}$.

Note that in free logic [8] $E!z$ corresponds to negation of εz.

Algebras of the form

$$ARE(V \cup U, A) =< Pr_A^{V\cup U}; \vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, \varepsilon z >$$

(for different A) constitute a semantic base for L_ε^{UR}.

A syntactic component is specified by the set of formulas $Fr(\Sigma_\varepsilon^U)$ where $\Sigma_\varepsilon^U = (V \cup U, \{\vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, \varepsilon z\}, Ps)$ is the signature of L_ε^{UR}.

An interpretational component of L_ε^{UR} is defined in the same way as for L^{UR}.

By construction of L_ε^{UR} we get the following statement.

Lemma 4. *L_ε^{UR} is a model-theoretic conservative extension of L^{UR}.*

Predicates εz specify cases when z is assigned or unassigned. This property can be used for construction of sequent rules for quantifiers.

For a formula Φ and a set of formulas Γ let $nm(\Phi)$ denote all names (variables) that occur in Φ, $nm(\Gamma)$ denote all names that occur in formulas of Γ. Names from $U \backslash nm(\Phi)$ are called fresh unessential variables for Φ and their set is denoted $fu(\Phi)$, in the same way $fu(\Gamma) = U \backslash nm(\Gamma)$ is the set of fresh unessential variables for Γ. We also use natural extensions of this notation for a case of several formulas and sets of formulas like $nm(\Gamma, \Delta, R_{\bar{v}}^{\bar{u}}(\exists x \Phi))$ and $fu(\Gamma, \Delta, R_{\bar{v}}^{\bar{u}}(\exists x \Phi))$. Such notation is also used when we consider properties of predicate algebras. We write $x \in \bar{v}$ to denote that x is a variable from $\bar{v}$. We write $\{\bar{v}, \bar{x}\}$ to denote the set of variables that occur in the sequences $\bar{v}$ and $\bar{x}$.

4 Semantic properties of quasiary logics

The set of compositions $\{\vee, \neg, R^{\bar{v}}_{\bar{x}}, \exists x, \varepsilon z\}$ of quasiary logics specifies four types of properties related to propositional compositions $\vee$ and $\neg$, to renomination composition $R^{\bar{v}}_{\bar{x}}$, to unassignment composition (predicate) εz, and to existential quantifier $\exists x$.

4.1 Properties related to propositional compositions

Properties of propositional compositions are traditional. We formulate here only one property of double negation

$$\neg\neg\colon\ \neg\neg p = p.$$

4.2 Properties related to renomination composition

Renomination composition is a new composition specific for logics of quasiary predicates. Its properties are not well-known therefore we describe them in more detail. The main attention will be paid to distributivity properties.

Preliminarily we define composition $R^{\bar{v}}_{\bar{x}} \circ^{\bar{w}}_{\bar{y}}$ which is the result of two successive renominations in the following way:

$$R^{v_1,...,v_n,w_1,...,w_m}_{x_1,...,x_n,y_1,...,y_m} \circ^{v_1,...,v_n,u_1,...,u_k}_{s_1,...,s_n,z_1,...,z_k} (p) =$$

$$= R^{v_1,...,v_n,w_1,...,w_m}_{x_1,...,x_n,y_1,...,y_m}(R^{v_1,...,v_n,u_1,...,u_k}_{s_1,...,s_n,z_1,...,z_k}(p)) =$$

$$= R^{v_1,...,v_n,w_1,...,w_m,u_1,...,u_k}_{\alpha_1,...,\alpha_n,y_1,...,y_m,\beta_1,...,\beta_k}(p),$$

where $w_i \neq u_j$ $(i = 1, ..., m; j = 1, ..., k)$,
$\alpha_i = s_i(v_1, ..., v_n, w_1, ..., w_m/x_1, ..., x_n, y_1, ..., y_m)$,
$\beta_j = z_j(v_1, ..., v_n, w_1, ..., w_m/x_1, ..., x_n, y_1, ..., y_m)$
Here

$$r(b_1..., b_q/c_1, ..., c_q) = r \text{ if } r \notin \{b_1, ..., b_q\},$$
$$r(b_1, ..., b_q/c_1, ..., c_q) = c_i \text{ if } r = b_i \text{ for some } i.$$

In the sequel we adopt the following convention: a, b denote elements from A; x, y, z, v, w (maybe with indexes) denote variables (names) from $V \cup U$; d, d', d_1, d_2 denote nominative sets from ${}^{V\cup U}A$; p, q

denote predicates from $ARE(V \cup U, A)$; Φ, Ψ, Ξ denote L_ε^{UR}-formulas, Γ, Δ denote sets of L_ε^{UR}-formulas, J denotes L_ε^{UR}-interpretation.

Lemma 5. *For every algebra $ARE(V \cup U, A)$ the following properties related to renomination composition hold:*

R∨: $R_{\bar{x}}^{\bar{v}}(p \vee q) = R_{\bar{x}}^{\bar{v}}(p) \vee R_{\bar{x}}^{\bar{v}}(q)$;
R¬: $R_{\bar{x}}^{\bar{v}}(\neg p) = \neg R_{\bar{x}}^{\bar{v}}(p)$;
RI: $R_{z,\bar{x}}^{z,\bar{v}}(p) = R_{\bar{x}}^{\bar{v}}(p)$;
RU: $R_{z,\bar{x}}^{y,\bar{v}}(p) = R_{\bar{x}}^{\bar{v}}(p)$, *$z$ is unessential for* $R_{\bar{x}}^{\bar{v}}(p)$;
RR: $R_{\bar{x}}^{\bar{v}}(R_{\bar{y}}^{\bar{w}}(p)) = R_{\bar{x}}^{\bar{v}} \circ_{\bar{y}}^{\bar{w}} (p)$;
R: $R(p) = p$;
R∃1: $R_{\bar{x}}^{\bar{v}}(\exists yp) = \exists y(R_{\bar{x}}^{\bar{v}}(p))$, $y \notin \{\bar{v}, \bar{x}\}$;
R∃2: $\exists yp = \exists z R_z^y(p)$, *$z$ is unessential for p;*
R∃3: $R_{z,\bar{x}}^{y,\bar{v}}(\exists yp) = R_{\bar{x}}^{\bar{v}}(\exists yp)$;
R∃4: $R_{\bar{x}}^{\bar{v}}(\exists yp) = \exists z R_{\bar{x}}^{\bar{v}}(R_z^y(p))$, *$z$ is unessential for* $R_{\bar{x}}^{\bar{v}}(\exists yp)$.

Proof. We prove the lemma by showing that truth and falsity domains for predicates in the left- and right-hand sides of equalities coincide. Let us consider properties R∃1, R∃2, and R∃4 only.

For R∃1 we have:

$d \in T(R_{\bar{x}}^{\bar{v}}(\exists yp)) \Leftrightarrow r_{\bar{x}}^{\bar{v}}(d) \in T(\exists yp) \Leftrightarrow r_{\bar{x}}^{\bar{v}}(d)\nabla y \mapsto a \in T(p)$ for some $a \in A \Leftrightarrow$ (since $y \notin \{\bar{v}, \bar{x}\}$) $r_{\bar{x}}^{\bar{v}}(d\nabla y \mapsto a) \in T(p)$ for some $a \in A \Leftrightarrow d\nabla y \mapsto a \in T(R_{\bar{x}}^{\bar{v}}(p))$ for some $a \in A \Leftrightarrow d \in T(\exists y(R_{\bar{x}}^{\bar{v}}(p)))$;

$d \in F(R_{\bar{x}}^{\bar{v}}(\exists yp)) \Leftrightarrow r_{\bar{x}}^{\bar{v}}(d) \in F(\exists yp) \Leftrightarrow r_{\bar{x}}^{\bar{v}}(d)\nabla y \mapsto a \in F(p)$ for all $a \in A \Leftrightarrow$ (since $y \notin \{\bar{v}, \bar{x}\}$) $r_{\bar{x}}^{\bar{v}}(d\nabla y \mapsto a) \in F(p)$ for all $a \in A \Leftrightarrow d\nabla y \mapsto a \in F(R_{\bar{x}}^{\bar{v}}(p))$ for all $a \in A \Leftrightarrow d \in F(\exists y(R_{\bar{x}}^{\bar{v}}(p)))$.

For R∃2 we have:

$d \in T(\exists z(R_z^y(p))) \Leftrightarrow d\nabla z \mapsto a \in T(R_z^y(p))$ for some $a \in A \Leftrightarrow r_z^y(d\nabla z \mapsto a) \in T(p)$ for some $a \in A \Leftrightarrow (d\nabla z \mapsto a)\nabla y \mapsto a \in T(p)$ for some $a \in A \Leftrightarrow$ (since z is unessential for p) $d\nabla y \mapsto a \in T(p)$ for some $a \in A \Leftrightarrow d \in T(\exists yp)$.

In the same way we demonstrate coincidence of the falsity domains for R∃2.

By R∃1 and R∃2 we obtain R∃4.

□

4.3 Properties related to unassignment composition

Here we formulate only that null-ary unassignment composition (predicate) is total deterministic predicate, i.e.

$$T(\varepsilon y) \cup F(\varepsilon y) = {}^{V}A \text{ and } T(\varepsilon y) \cap F(\varepsilon y) = \emptyset.$$

4.4 Properties related to quantifier composition

The following lemma describes some properties of quantifiers.

Lemma 6. *For every algebra $ARE(V \cup U, A)$ the following properties related to quantifiers hold ($x \neq y$):*

$$T\exists_R\text{: } T(R^x_y(p)) \cap F(\varepsilon y) \subseteq T(\exists x p);$$
$$T\exists_L\text{: } T(\exists x p)|_{-y} \subseteq (T(R^x_y(p)) \cap F(\varepsilon y))|_{-y} \text{ if } y \textit{ is unessential for } p.$$

Proof. For the first property let $d \in T(R^x_y(p)) \cap F(\varepsilon y)$, then 1) there exist $a \in A$ such that $y \mapsto a \in_n d$ and 2) $d\nabla x \mapsto a \in T(p)$. Therefore $d \in T(\exists x p)$. Thus, $T\exists_R$ holds.

For the second property let $d \in T(\exists x p)|_{-y}$. It means that there exists $d' \in {}^{V\cup U}A$ and $a \in A$ such that $d'\nabla x \mapsto a \in T(p)$ and $d' =_{-y} d$. Since y is not essential for p, then $(d'\nabla x \mapsto a)\nabla y \mapsto a \in T(p)$. By definition, we get that $d'\nabla y \mapsto a \in T(R^x_y(p)) \cap F(\varepsilon y)$. But $d = d'|_{-y}$. This proves $T\exists_L$.

□

Now we can formulate semantic properties for quantifiers.

Lemma 7. *For every algebra $ARE(V \cup U, A)$ the following properties related to quantifiers hold ($x \neq y$):*

$$\exists e_L\text{: } T(\exists x p) =_{-y} (T(R^x_y(p)) \cap F(\varepsilon y)) \text{ if } y \textit{ is unessential for } p;$$
$$\exists e_R\text{: } T(\exists x p) \cup T(\varepsilon y) = T(R^x_y(p)) \cup T(\exists x p) \cup T(\varepsilon y).$$

Proof. The first property follows by $T\exists_R$ and $T\exists_L$. For the second property, inclusion $T(\exists x p) \cup T(\varepsilon y) \subseteq T(R^x_y(p)) \cup T(\exists x p) \cup T(\varepsilon y)$ is obvious. Inclusion $T(\exists x p) \cup T(\varepsilon y) \supseteq T(R^x_y(p)) \cup T(\exists x p) \cup T(\varepsilon y)$ follows by $T\exists_R$ which can be presented as $T(R^x_y(p)) \subseteq T(\varepsilon y) \cup T(\exists x p)$. To get such presentation we use the following property of Boolean algebra of

sets: $S_1 \cap S_2 \subseteq S_3 \Leftrightarrow S_1 \subseteq \overline{S_2} \cup S_3$ where $\overline{S_2}$ denotes supplement of S_2 and the properties that $\overline{T(\varepsilon y)} = F(\varepsilon y)$ and $\overline{F(\varepsilon y)} = T(\varepsilon y)$.

□

5 Consequence relations for sets of formulas

We consider two consequence relations: conventional logical consequence and special T-consequence.

Let $\Gamma \subseteq Fr(\Sigma_\varepsilon^U)$ and $\Delta \subseteq Fr(\Sigma_\varepsilon^U)$ be sets of formulas. Δ is a *consequence* of Γ in an interpretation J (denoted $\Gamma_J \models \Delta$), if $\bigcap\limits_{\Phi\in\Gamma} T(\Phi_J) \cap \bigcap\limits_{\Psi\in\Delta} F(\Psi_J) = \emptyset$.

Δ is a logical consequence of Γ (denoted $\Gamma \models \Delta$), if $\Gamma_J \models \Delta$ in every interpretation J. The introduced relation of logical consequence specifies irrefutability.

For the class of non-deterministic predicates the logical consequence relation collapses, i.e. it is empty. Indeed, for any Γ and Δ we have that $\Gamma_J \not\models \Delta$ if we interpret predicate symbols as non-deterministic predicate $\top_A^V$ (Lemma 2).

Therefore we introduce another consequence relation which arises naturally in Computer Science [9].

Δ is a *T-consequence* of Γ in an interpretation J (denoted by $\Gamma_J \models_T \Delta$), if $\bigcap\limits_{\Phi\in\Gamma} T(\Phi_J) \subseteq \bigcup\limits_{\Psi\in\Delta} T(\Psi_J)$. Δ is a T-consequence of Γ (denoted by $\Gamma \models_T \Delta$), if $\Gamma_J \models_T \Delta$ in every interpretation J.

We will also use the following notation: $T^\wedge(\Gamma_J) = \bigcap\limits_{\Phi\in\Gamma} T(\Phi_J)$ and $T^\vee(\Gamma_J) = \bigcup\limits_{\Phi\in\Gamma} T(\Phi_J)$.

Now we describe the main properties of T-consequence relation.

First, let us give the following definitions for arbitrary consequence relation $\models_*$ [10]:

- $\models_*$ is called *paraconsistent* if there exist Γ, Φ, and Ψ such that $\Gamma, \Phi \wedge \neg\Phi \not\models_* \Psi$;
- $\models_*$ is called *paracomplete* if there exist Γ, Φ, and Ψ such that $\Gamma, \Phi \not\models_* \Psi \vee \neg\Psi$;

– $\models_*$ is called *paranormal* if there exist Γ, Φ, and Ψ such that $\Gamma, \Phi \wedge \neg\Phi \not\models_* \Psi \vee \neg\Psi$.

Lemma 8. *T-consequence relation is paraconsistent, paracomplete, and paranormal.*

Proof. We demonstrate only paranormality of T-consequence relation. Indeed, let $\Gamma = \emptyset$, Φ be $P_1 \in Ps$, Ψ be $P_2 \in Ps$ such that $P_1 \neq P_2$. Then it is easy to check that interpreting P_1 as predicate $\top_A^V$ and P_2 as predicate $\bot_A^V$ we get that $P_1 \wedge \neg P_1 \not\models_T P_2 \vee \neg P_2$. □

In a similar way we can demonstrate that the following statement holds.

Lemma 9. *There exist Γ, Δ, Φ, and Ψ such that*

– $\Gamma, \neg\Phi \models_T \Delta$ *and* $\Gamma \not\models_T \Phi, \Delta$*;*

– $\Gamma, \Phi \models_T \Delta$ *and* $\Gamma \not\models_T \neg\Phi, \Delta$*;*

– $\Gamma \models_T \neg\Phi, \Delta$ *and* $\Gamma, \Phi \not\models_T \Delta$*;*

– $\Gamma \models_T \Phi, \Delta$ *and* $\Gamma, \neg\Phi \not\models_T \Delta$.

This lemma states that rules of sequent calculi permitting moving (negated) formulas from one side of a sequent to its another side are not valid for T-consequence relations. Consequently, sequent calculi for $\models_T$ will be more complicated.

Still, such transformations are possible for a formula interpreted as total deterministic predicate (see Lemma 11 (4)).

Lemma 10. *Let Φ be a formula, $\Gamma, \Gamma', \Delta, \Delta'$ be sets of formulas. Then*

(M) if $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$, then $\Gamma \models_T \Delta \Rightarrow \Gamma' \models_T \Delta'$;

(C) $\Phi, \Gamma \models_T \Delta, \Phi$.

Proof of the lemma follows immediately from definitions.

Now we continue with those properties of T-consequence relation which induce sequent rules for the logic under consideration. Such properties are constructed upon semantic properties of compositions. To do this the following lemma is often used.

Lemma 11. *Let Φ, Ψ, and Ξ be formulas, Γ and Δ be sets of formulas, J be L_ε^{UR}-interpretation. Then*

(1) if $T(\Phi_J) = T(\Psi_J)$, then
$$\Phi, \Gamma\ {}_J\models_T \Delta \Leftrightarrow \Psi, \Gamma\ {}_J\models_T \Delta \text{ and } \Gamma\ {}_J\models_T \Phi, \Delta \Leftrightarrow \Gamma\ {}_J\models_T \Psi, \Delta;$$

(2) if $T(\Phi_J) = T(\Psi_J) \cap T(\Xi_J)$, then
$$\Phi, \Gamma\ {}_J\models_T \Delta \Leftrightarrow \Psi, \Xi, \Gamma\ {}_J\models_T \Delta,$$
$$\Gamma\ {}_J\models_T \Phi, \Delta \Leftrightarrow (\Gamma\ {}_J\models_T \Psi, \Delta \text{ and } \Gamma\ {}_J\models_T \Xi, \Delta);$$

(3) if $T(\Phi_J) = T(\Psi_J) \cup T(\Xi_J)$, then
$$\Gamma\ {}_J\models_T \Phi, \Delta \Leftrightarrow \Gamma\ {}_J\models_T \Psi, \Xi, \Delta,$$
$$\Phi, \Gamma\ {}_J\models_T \Delta \Leftrightarrow (\Psi, \Gamma\ {}_J\models_T \Delta \text{ and } \Xi, \Gamma\ {}_J\models_T \Delta);$$

(4) if $T(\Phi_J) \cup F(\Phi_J) = {}^V\!A$ and $T(\Phi_J) \cap F(\Phi_J) = \emptyset$, then
$$\Phi, \Gamma\ {}_J\models_T \Delta \Leftrightarrow \Gamma\ {}_J\models_T \neg\Phi, \Delta \text{ and } \neg\Phi, \Gamma\ {}_J\models_T \Delta \Leftrightarrow \Gamma\ {}_J\models_T \Phi, \Delta;$$
$$\Gamma \models_T \Delta \Leftrightarrow (\Phi, \Gamma \models_T \Delta \text{ and } \Gamma \models_T \Delta, \Phi);$$

(5) if $y \in fu(\Gamma, \Delta)$, then $\Gamma\ {}_J\models_T \Delta \Leftrightarrow \Gamma\ {}_J\models_T \Delta, \varepsilon y$;

(6) if $T(\Phi_J) =_{-y} T(\Psi_J)$ for $y \in fu(\Psi, \Gamma, \Delta)$, then
$$\Phi, \Gamma\ {}_J\models_T \Delta \Leftrightarrow \Psi, \Gamma\ {}_J\models_T \Delta.$$

Proof. Property (1) is obvious. For (2) we have
$\Phi, \Gamma\ {}_J\models_T \Delta \Leftrightarrow T(\Phi_J) \cap T^\wedge(\Gamma_J) \subseteq T^\vee(\Delta_J) \Leftrightarrow$
$\Leftrightarrow T(\Psi_J) \cap T(\Xi_J) \cap T^\wedge(\Gamma_J) \subseteq T^\vee(\Delta_J) \Leftrightarrow \Psi, \Xi, \Gamma\ {}_J\models_T \Delta.$

In the same way the second part of (2) and property (3) are proved. Let us consider (4). We have
$\Phi, \Gamma\ {}_J\models_T \Delta \Leftrightarrow \overline{T(\Phi_J)} \cap T^\wedge(\Gamma_J) \subseteq T^\vee(\Delta_J) \Leftrightarrow$
$\Leftrightarrow T^\wedge(\Gamma_J) \subseteq \overline{T(\Phi_J)} \cup T^\vee(\Delta_J) \Leftrightarrow T^\wedge(\Gamma_J) \subseteq T(\neg\Phi_J) \cup T^\vee(\Delta_J) \Leftrightarrow$
$\Leftrightarrow \Gamma\ {}_J\models_T \neg\Phi, \Delta.$

In the same way other properties of (4) are proved.

Let us consider (5). By Lemma 10(M) we have that $\Gamma\ {}_J\models_T \Delta \Rightarrow \Gamma\ {}_J\models_T \Delta, \varepsilon y$. We need to prove that $\Gamma\ {}_J\models_T \Delta, \varepsilon y \Rightarrow \Gamma\ {}_J\models_T \Delta$. It is equivalent to
$T^\wedge(\Gamma_J) \subseteq T^\vee(\Delta_J) \cup T(\varepsilon y) \Leftrightarrow T^\wedge(\Gamma_J) \cap F(\varepsilon y) \subseteq T^\vee(\Delta_J)$.

Let $d \in T^\wedge(\Gamma_J) \cap F(\varepsilon y)$. Since y is unessential for Γ it means that $d|_{-y} \in T^\wedge(\Gamma_J)$. From this follows that $d|_{-y} \in T^\vee(\Delta_J)$. Since y is

unessential for Δ it means that $d \in T^{\vee}(\Delta_J)$. Thus, $T^{\wedge}(\Gamma_J) \subseteq T^{\vee}(\Delta_J)$ that proves the property under consideration.

Let us consider (6). We should prove that

$$T(\Phi_J) \cap T^{\wedge}(\Gamma_J) \subseteq T^{\vee}(\Delta_J) \Leftrightarrow T(\Psi_J) \cap T^{\wedge}(\Gamma_J) \subseteq T^{\vee}(\Delta_J).$$

Let $d \in T(\Psi_J) \cap T^{\wedge}(\Gamma_J)$. Since $T(\Phi_J) =_{-y} T(\Psi_J)$ there exists $d' \in T(\Phi_J)$ such that $d' =_{-y} d$. Since y is unessential for Γ we have that $d' \in T^{\wedge}(\Gamma_J)$. Hence $d' \in T^{\vee}(\Delta_J)$. Again, y is also unessential for Δ therefore $d \in T^{\vee}(\Delta_J)$. This proves the direct implication.

Let us prove the inverse implication. First, we prove that $T(\Phi_J) \subseteq T(\Psi_J)$. Indeed, let $d \in T(\Phi_J)$. Since $T(\Phi_J) =_{-y} T(\Psi_J)$ there exists $d' \in T(\Psi_J)$ such that $d' =_{-y} d$. Since y is unessential for Ψ, $d \in T(\Psi_J)$. Thus, $T(\Phi) \subseteq T(\Psi)$.

From this follows that $\Psi, \Gamma\,{}_J\!\models_T \Delta \Rightarrow \Phi, \Gamma\,{}_J\!\models_T \Delta$.

This completes the proof of (6).

□

Theorem 1. *The following properties hold for T-consequence relation.*

– *Properties related to propositional compositions:*

$\neg\neg_{\mathrm{L}})$ $\neg\neg\Phi, \Gamma \models_T \Delta \Leftrightarrow \Phi, \Gamma \models_T \Delta$.

$\neg\neg_{\mathrm{R}})$ $\Gamma \models_T \Delta, \neg\neg\Phi \Leftrightarrow \Gamma \models_T \Delta, \Phi$.

$\vee_{\mathrm{L}})$ $\Phi \vee \Psi, \Gamma \models_T \Delta \Leftrightarrow (\Phi, \Gamma \models_T \Delta$ *and* $\Psi, \Gamma \models_T \Delta)$.

$\neg \vee_{\mathrm{L}})$ $\neg(\Phi \vee \Psi), \Gamma \models_T \Delta \Leftrightarrow \neg\Phi, \neg\Psi, \Gamma \models_T \Delta$.

$\vee_{\mathrm{R}})$ $\Gamma \models_T \Delta, \Phi \vee \Psi \Leftrightarrow \Gamma \models_T \Delta, \Phi, \Psi$.

$\neg \vee_{\mathrm{R}})$ $\Gamma \models_T \Delta, \neg(\Phi \vee \Psi) \Leftrightarrow (\Gamma \models_T \Delta, \neg\Phi$ *and* $\Gamma \models_T \Delta, \neg\Psi)$.

– *Properties related to renomination compositions:*

$R \vee_{\mathrm{L}})$ $R^{\bar{v}}_{\bar{x}}(\Phi \vee \Psi), \Gamma \models_T \Delta \Leftrightarrow R^{\bar{v}}_{\bar{x}}(\Phi) \vee R^{\bar{v}}_{\bar{x}}(\Psi), \Gamma \models_T \Delta$.

$\neg R \vee_{\mathrm{L}})$ $\neg R^{\bar{v}}_{\bar{x}}(\Phi \vee \Psi), \Gamma \models_T \Delta \Leftrightarrow \neg(R^{\bar{v}}_{\bar{x}}(\Phi) \vee R^{\bar{v}}_{\bar{x}}(\Psi)), \Gamma \models_T \Delta$.

$R \vee_{\mathrm{R}})$ $\Gamma \models_T \Delta, R^{\bar{v}}_{\bar{x}}(\Phi \vee \Psi) \Leftrightarrow \Gamma \models_T \Delta, R^{\bar{v}}_{\bar{x}}(\Phi) \vee R^{\bar{v}}_{\bar{x}}(\Psi)$.

$\neg R \vee_{\mathrm{R}})$ $\Gamma \models_T \Delta, \neg R^{\bar{v}}_{\bar{x}}(\Phi \vee \Psi) \Leftrightarrow \Gamma \models_T \Delta, \neg(R^{\bar{v}}_{\bar{x}}(\Phi) \vee R^{\bar{v}}_{\bar{x}}(\Psi))$.

$\mathrm{R}_{\mathrm{L}})$ $R(\Phi), \Gamma \models_T \Delta \Leftrightarrow \Phi, \Gamma \models_T \Delta$.

$\mathrm{R_R})\ \Gamma \models_T \Delta, R(\Phi) \Leftrightarrow \Phi, \Gamma \models_T \Delta, \Phi.$

$\mathrm{RI_L})\ R^{z,\bar{v}}_{z,\bar{x}}(\Phi), \Gamma \models_T \Delta \Leftrightarrow R^{\bar{v}}_{\bar{x}}\Phi), \Gamma \models_T \Delta.$

$\neg\mathrm{RI_L})\ \neg R^{z,\bar{v}}_{z,\bar{x}}(\Phi), \Gamma \models_T \Delta \Leftrightarrow \neg R^{\bar{v}}_{\bar{x}}\Phi), \Gamma \models_T \Delta.$

$\mathrm{RI_R})\ \Gamma \models_T \Delta, R^{z,\bar{v}}_{z,\bar{x}}(\Phi) \Leftrightarrow \Gamma \models_T \Delta, R^{\bar{v}}_{\bar{x}}\Phi).$

$\neg\mathrm{RI_R})\ \Gamma \models_T \Delta, \neg R^{z,\bar{v}}_{z,\bar{x}}(\Phi) \Leftrightarrow \Gamma \models_T \Delta, \neg R^{\bar{v}}_{\bar{x}}\Phi).$

$\mathrm{RU_L})\ R^{y,\bar{v}}_{z,\bar{x}}(\Phi), \Gamma \models_T \Delta \Leftrightarrow R^{\bar{v}}_{\bar{x}}(\Phi), \Gamma \models_T \Delta,\ \textit{if}\ y \in fu(\Phi).$

$\neg\mathrm{RU_L})\ \neg R^{y,\bar{v}}_{z,\bar{x}}(\Phi), \Gamma \models_T \Delta \Leftrightarrow \neg R^{\bar{v}}_{\bar{x}}(\Phi), \Gamma \models_T \Delta,\ \textit{if}\ y \in fu(\Phi).$

$\mathrm{RU_R})\ \Gamma \models_T \Delta, R^{y,\bar{v}}_{z,\bar{x}}(\Phi) \Leftrightarrow \Gamma \models_T \Delta, R^{\bar{v}}_{\bar{x}}(\Phi),\ \textit{if}\ y \in fu(\Phi).$

$\neg\mathrm{RU_R})\ \Gamma \models_T \Delta, \neg R^{y,\bar{v}}_{z,\bar{x}}(\Phi), \Leftrightarrow \Gamma \models_T \Delta, \neg R^{\bar{v}}_{\bar{x}}(\Phi),\ \textit{if}\ y \in fu(\Phi).$

$\mathrm{RR_L})\ R^{\bar{v}}_{\bar{x}}(R^{\bar{w}}_{\bar{y}}(\Phi)), \Gamma \models_T \Delta \Leftrightarrow R^{\bar{v}}_{\bar{x}} \circ^{\bar{w}}_{\bar{y}} (\Phi), \Gamma \models_T \Delta.$

$\neg\mathrm{RR_L})\ \neg R^{\bar{v}}_{\bar{x}}(R^{\bar{w}}_{\bar{y}}(\Phi)), \Gamma \models_T \Delta \Leftrightarrow \neg R^{\bar{v}}_{\bar{x}} \circ^{\bar{w}}_{\bar{y}} (\Phi), \Gamma \models_T \Delta.$

$\mathrm{RR_R})\ \Gamma \models_T \Delta, R^{\bar{v}}_{\bar{x}}(R^{\bar{w}}_{\bar{y}}(\Phi)) \Leftrightarrow \Gamma \models_T \Delta, R^{\bar{v}}_{\bar{x}} \circ^{\bar{w}}_{\bar{y}} (\Phi).$

$\neg\mathrm{RR_R})\ \Gamma \models_T \Delta, \neg R^{\bar{v}}_{\bar{x}}(R^{\bar{w}}_{\bar{y}}(\Phi)) \Leftrightarrow \Gamma \models_T \Delta, \neg R^{\bar{v}}_{\bar{x}} \circ^{\bar{w}}_{\bar{y}} (\Phi).$

$\mathrm{R\neg_L})\ R^{\bar{v}}_{\bar{x}}(\neg\Phi)), \Gamma \models_T \Delta \Leftrightarrow \neg R^{\bar{v}}_{\bar{x}}(\Phi), \Gamma \models_T \Delta.$

$\neg\mathrm{R\neg_L})\ \neg R^{\bar{v}}_{\bar{x}}(\neg\Phi)), \Gamma \models_T \Delta \Leftrightarrow \neg\neg R^{\bar{v}}_{\bar{x}}(\Phi), \Gamma \models_T \Delta.$

$\mathrm{R\neg_R})\ \Gamma \models_T \Delta, R^{\bar{v}}_{\bar{x}}(\neg\Phi)) \Leftrightarrow \Gamma \models_T \Delta, \neg R^{\bar{v}}_{\bar{x}}(\Phi).$

$\neg\mathrm{R\neg_R})\ \Gamma \models_T \Delta, \neg R^{\bar{v}}_{\bar{x}}(\neg\Phi)) \Leftrightarrow \Gamma \models_T \Delta, \neg\neg R^{\bar{v}}_{\bar{x}}(\Phi).$

$\mathrm{R\exists R_L})\ R^{\bar{u},x}_{\bar{v},y}(\exists x\Phi), \Gamma \models_T \Delta \Leftrightarrow R^{\bar{u}}_{\bar{v}}(\exists x\Phi), \Gamma \models_T \Delta,\ \textit{if}\ x \notin \bar{u}.$

$\neg\mathrm{R\exists R_L})\ \neg R^{\bar{u},x}_{\bar{v},y}(\exists x\Phi), \Gamma \models_T \Delta \Leftrightarrow \neg R^{\bar{u}}_{\bar{v}}(\exists x\Phi), \Gamma \models_T \Delta,\ \textit{if}\ x \notin \bar{u}.$

$\mathrm{R\exists R_R})\ \Gamma \models_T \Delta, R^{\bar{u},x}_{\bar{v},y}(\exists x\Phi) \Leftrightarrow \Gamma \models_T \Delta, R^{\bar{u}}_{\bar{v}}(\exists x\Phi),\ \textit{if}\ x \notin \bar{u}.$

$\neg\mathrm{R\exists R_R})\ \Gamma \models_T \Delta, \neg R^{\bar{u},x}_{\bar{v},y}(\exists x\Phi) \Leftrightarrow \Gamma \models_T \Delta, \neg R^{\bar{u}}_{\bar{v}}(\exists x\Phi),\ \textit{if}\ x \notin \bar{u}.$

$\mathrm{R\exists_L})\ R^{\bar{v}}_{\bar{x}}(\exists y\Phi), \Gamma \models_T \Delta \Leftrightarrow \exists y R^{\bar{v}}_{\bar{x}}(\Phi), \Gamma \models_T \Delta,\ \textit{if}\ y \notin \{\bar{v}, \bar{x}\}.$

$\neg\mathrm{R\exists_L})\ \neg R^{\bar{v}}_{\bar{x}}(\exists y\Phi), \Gamma \models_T \Delta \Leftrightarrow \exists y \neg R^{\bar{v}}_{\bar{x}}(\Phi), \Gamma \models_T \Delta,\ \textit{if}\ y \notin \{\bar{v}, \bar{x}\}.$

$\mathrm{R\exists_R})\ \Gamma \models_T \Delta, R^{\bar{v}}_{\bar{x}}(\exists y\Phi) \Leftrightarrow \Gamma \models_T \Delta, \exists y R^{\bar{v}}_{\bar{x}}(\Phi),\ \textit{if}\ y \notin \{\bar{v}, \bar{x}\}.$

$\neg\mathrm{R\exists_R})\ \Gamma \models_T \Delta, \neg R^{\bar{v}}_{\bar{x}}(\exists y\Phi) \Leftrightarrow \Gamma \models_T \Delta, \exists y \neg R^{\bar{v}}_{\bar{x}}(\Phi),\ \textit{if}\ y \notin \{\bar{v}, \bar{x}\}.$

$\mathrm{R}\exists\exists_{\mathrm{L}})$ $R^{\bar{v}}_{\bar{x}}(\exists y\Phi)), \Gamma \models_T \Delta \Leftrightarrow \exists z R^{\bar{v}}_{\bar{x}} \circ^y_z (\Phi), \Gamma \models_T \Delta,$
$\textit{if } z \in fu(R^{\bar{v}}_{\bar{x}}(\exists y\Phi)).$

$\neg\mathrm{R}\exists\exists_{\mathrm{L}})$ $\neg R^{\bar{v}}_{\bar{x}}(\exists y\Phi)), \Gamma \models_T \Delta \Leftrightarrow \neg\exists z R^{\bar{v}}_{\bar{x}} \circ^y_z (\Phi), \Gamma \models_T \Delta,$
$\textit{if } z \in fu(R^{\bar{v}}_{\bar{x}}(\exists y\Phi)).$

$\mathrm{R}\exists\exists_{\mathrm{R}})$ $\Gamma \models_T \Delta, R^{\bar{v}}_{\bar{x}}(\exists y\Phi)) \Leftrightarrow \Gamma \models_T \Delta, \exists z R^{\bar{v}}_{\bar{x}} \circ^y_z (\Phi),$
$\textit{if } z \in fu(R^{\bar{v}}_{\bar{x}}(\exists y\Phi)).$

$\neg\mathrm{R}\exists\exists_{\mathrm{R}})$ $\Gamma \models_T \Delta, \neg R^{\bar{v}}_{\bar{x}}(\exists y\Phi)) \Leftrightarrow \Gamma \models_T \Delta, \neg\exists z R^{\bar{v}}_{\bar{x}} \circ^y_z (\Phi),$
$\textit{if } z \in fu(R^{\bar{v}}_{\bar{x}}(\exists y\Phi)).$

– *Properties related to unassignment predicates:*

$\varepsilon_{\mathrm{LR}})$ $\Gamma \models_T \Delta \Leftrightarrow \varepsilon y, \Gamma \models_T \Delta$ *and* $\Gamma \models_T \Delta, \varepsilon y.$

$\varepsilon_{\mathrm{R}})$ $\Gamma \models_T \Delta \Leftrightarrow \Gamma \models_T \Delta, \varepsilon z,$ *if* $z \in fu(\Gamma, \Delta).$

– *Properties related to quantifiers:*

$\exists\mathrm{Re}_{\mathrm{L}})$ $R^{\bar{u}}_{\bar{v}}(\exists x\Phi), \Gamma \models_T \Delta \Leftrightarrow R^{\bar{u},x}_{\bar{v},z}(\Phi), \Gamma \models_T \Delta, \varepsilon z,$
$\textit{if } z \in fu(\Gamma, \Delta, R^{\bar{u}}_{\bar{v}}(\exists x\Phi)).$

$\neg\exists\mathrm{Re}_{\mathrm{L}})$ $\neg R^{\bar{u}}_{\bar{v}}(\exists x\Phi), \Gamma \models_T \Delta, \varepsilon y \Leftrightarrow$
$\Leftrightarrow \neg R^{\bar{u}}_{\bar{v}}(\exists x\Phi), \neg R^{\bar{u},x}_{\bar{v},y}(\Phi), \Gamma \models_T \Delta, \varepsilon y.$

$\exists\mathrm{Re}_{\mathrm{R}})$ $\Gamma \models_T \Delta, R^{\bar{u}}_{\bar{v}}(\exists x\Phi), \varepsilon y \Leftrightarrow \Gamma \models_T \Delta, R^{\bar{u}}_{\bar{v}}(\exists x\Phi), R^{\bar{u},x}_{\bar{v},y}(\Phi), \varepsilon y.$

$\neg\exists\mathrm{Re}_{\mathrm{R}})$ $\Gamma \models_T \Delta, \neg R^{\bar{u}}_{\bar{v}}(\exists x\Phi), \Leftrightarrow \Gamma \models_T \Delta, \neg R^{\bar{u},x}_{\bar{v},z}(\Phi), \varepsilon z,$
$\textit{if } z \in fu(\Gamma, \Delta, R^{\bar{u}}_{\bar{v}}(\exists x\Phi)).$

Proof. Proof of the formulated properties is based on semantic properties of compositions and properties of T-consequence relation. For example, $\exists\mathrm{Re}_{\mathrm{L}})$ and $\neg\exists\mathrm{Re}_{\mathrm{R}})$ are consequences of Lemmas 7($\exists e_L$), 11(6); $\neg\exists\mathrm{Re}_{\mathrm{L}})$ and $\exists\mathrm{Re}_{\mathrm{R}})$ are consequences of Lemmas 7($\exists e_R$), 11(6). □

6 Conclusion

In the paper we have investigated a special kind of program-oriented algebras and logics defined for classes of non-deterministic quasiary predicates. We have presented semantic properties related to propositional compositions, renomination, and quantifier compositions. These

properties form a basis for construction of sequent calculi for logics of non-deterministic quasiary predicates. We plan to present such calculi and prove their completeness in forthcoming papers.

References

[1] *Handbook of Logic in Computer Science*, S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum (eds.), in 5 volumes, Oxford Univ. Press, Oxford, 1993-2001.

[2] *Handbook of Philosophical Logic*, D.M. Gabbay, F. Guenthner (eds.), 2nd Edition, in 17 volumes, Springer, 2001–2011.

[3] M. Nikitchenko, S. Shkilniak. *Mathematical logic and theory of algorithms*, Publishing house of Taras Shevchenko National University of Kyiv, Kyiv, 2008, 528 p. (In Ukrainian)

[4] M. Nikitchenko, S. Shkilniak. *Applied Logic*, Publishing house of Taras Shevchenko National University of Kyiv, Kyiv, 2013, 278 p. (in Ukrainian).

[5] M. Nikitchenko, V. Tymofieiev. *Satisfiability in composition-nominative logics*, Central European Journal of Computer Science, vol. 2, no. 3, 2012, pp. 194–213.

[6] N. Nikitchenko. *A Composition Nominative Approach to Program Semantics*, Technical Report IT-TR 1998-020, Technical University of Denmark, 1998.

[7] A. Avron, A. Zamansky. *Non-deterministic semantics for logical systems*, in *Handbook of Philosophical Logic*, D.M. Gabbay, F. Guenthner (eds.), 2nd ed., vol. 16, Springer Netherlands, 2011, pp. 227–304.

[8] E. Bencivenga. *Free Logics*, in *Handbook of Philosophical Logic*, D. Gabbay and F. Guenthner (eds.), vol. III: Alternatives to Classical Logic, Dordrecht: D. Reidel, 1986, pp. 373–426.

[9] A. Kryvolap, M. Nikitchenko, W. Schreiner. *Extending Floyd-Hoare logic for partial pre- and postconditions*, CCIS, 412, 2013, Springer, Heidelberg, pp. 355-378.

[10] J.-Y. Béziau. *What is paraconsistent logic?*, in D. Batens, C. Mortensen, G. Priest, J.P. Van Bendegem (Eds.), Frontiers of Paraconsistent Logic, Proceedings of the 1st World Congress on Paraconsistency, held in Ghent, Belgium, July 29–August 3, 1997, Research Studies Press, Baldock, UK, 2000, pp. 95–111.

Mykola Nikitchenko, Stepan Shkilniak, Received July 19, 2015

Mykola Nikitchenko
Taras Shevchenko National University of Kyiv
01601, Kyiv, Volodymyrska st, 60
Phone: +38044 2590519
E–mail: nikitchenko@unicyb.kiev.ua

Stepan Shkilniak
Taras Shevchenko National University of Kyiv
01601, Kyiv, Volodymyrska st, 60
Phone: +38044 2590519
E–mail: sssh@unicyb.kiev.ua

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Modal Logics of Partial Predicates without Monotonicity Restriction

Oksana Shkilniak

Abstract

Modal logics recently have found many applications in various fields, including theoretical and applied computer science, philosophy and linguistics ([1,2]). Traditional modal logics are usually based on classical predicate logic. However, classical logic has some fundamental restrictions which don't allow taking into account sufficiently incompleteness, partiality and uncertainty of information. Composition-nominative logics of partial quasiary predicates (see [3,4]) is a program-oriented logical formalism based on wide classes of partial mappings over nominative data. Composition-nominative modal logics (CNML) combine facilities of traditional modal logics and composition-nominative logics. Their important variant, modal transitional logics (MTL), can adequately represent the fact of changing and evolution in subject domains. Traditional modal logics (alethic, temporal, deontic etc) can be easily considered within a scope of MTL. In this paper we introduce pure first-order MTL of partial quasiary predicates without monotonicity restriction. We define languages and semantic models of pure first-order MTL of non-monotone predicates and investigate their semantic properties, including properties of logical consequence relations for sets of formulas, specified with states. We distinguish various classes of MTL: multimodal, temporal, epistemic and general MTL. Significant difference between MTL of monotone and non-monotone predicates is demonstrated. Properties of logical consequence relations for sets of formulas, specified with states, are considered. Basing on these properties, corresponding sequent calculi can be constructed.

Keywords: Modal logic, partial predicate, non-monotone predicate, logical consequence.

1 Composition Nominative Modal Systems

Composition nominative modal system (CNMS) is the main semantic notion in composition nominative modal logic (CNML). Such systems describe possible worlds of modal logic and are their models.

We specify CNMS as an object $\mathbf{M} = (Cms, Fm, Im)$, where

- Cms is a composition modal system (CMS), which defines semantic aspects of the world;
- Fm is a set of formulas of the modal language;
- Im is an interpretation mapping of formulas on states of the universe.

CMS are relational semantic models: $Cms = (S, R, Pr, C)$, where S is a set of states of the universe, R is a set of relations on states $\rho \subseteq S \times S^n$, Pr is a set of predicates over state data, C is a set of compositions on predicates.

Let us refine the description of CNMS for pure first-order CNML. We define S as a set of algebraic structures $\alpha = (A_\alpha, Pr_\alpha)$, where A_α is a set of basic data of state α, Pr_α is a set of quasiary predicates ${}^V A_\alpha \to \{T, F\}$ (predicates of state α). $A = \bigcup\limits_{\alpha \in S} A_\alpha$ is a set of basic data of the world. Let us call predicates of the type ${}^V A \to \{T, F\}$ global.

For better understanding it will be useful to give some **basic notions** [4]. Quasiary predicate is a predicate over nominative sets (sets of pairs "data–value"). *V-nominative set on A* (V-NS) is an arbitrary single-valued function $\delta : V \to A$. We consider V as a set of names (variables) and A as a set of values (basic data).

V-NS can be presented as $[v_1 \mapsto a_1, \dots, v_n \mapsto a_n, \dots]$, where $v_i \in V$, $a_i \in A$ and $v_i \neq v_j$ when $i \neq j$. Set of all V-NS on A will be denoted as ${}^V A$.

Let us define function $asn : {}^V A \to 2^V$ as

$$asn(d) = \{v \in V \mid v \mapsto a \in d \text{ for some } a \in A\}.$$

We specify operation $||_{-x}$ of deletion of the component with name x for V-NS: $d||_{-x} = \{v \mapsto a \in d \mid v \neq x\}$.

Operation ∇ is defined as follows:

$$d_1 \nabla d_2 = d_2 \cup \{v \mapsto d_1 \mid v \notin asn(d_2)\}.$$

The definition of operation of renomination $\mathbf{r}^{v_1,\ldots,v_n}_{x_1,\ldots,x_n} : {}^V A \to {}^V A$:

$$\mathbf{r}^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}(d) = d\nabla[v_1 \mapsto d(x_1), \ldots, v_n \mapsto d(x_n)].$$

We can shorten $y_1, \ldots, y_n$ to $\bar{y}$, so instead of $\mathbf{r}^{v_1,\ldots,v_n}_{x_1,\ldots,x_n}$ we can write $\mathbf{r}^{\bar{v}}_{\bar{x}}$. Consecutive application of two operations of renomination $\mathbf{r}^{\bar{v}}_{\bar{x}}$ and $\mathbf{r}^{\bar{u}}_{\bar{y}}$ can be presented as a convolution of renominations $\mathbf{r}^{\bar{v}}_{\bar{x}} \centerdot \mathbf{r}^{\bar{u}}_{\bar{y}}$ (see [4]).

V-quasiary predicate on A [4] is an arbitrary (partial) function of the type ${}^V A \to \{T, F\}$, where T and F are truth values **true** and **false**.

Let us call the name $x \in V$ *unessential* for a quasiary predicate P if for arbitrary $d_1, d_2 \in {}^V A$ we have

$$d_1||_{-x} = d_2||_{-x} \Rightarrow P(d_1) = P(d_2).$$

Quasiary predicate P we call *monotone* (equitone) if for arbitrary $d_1, d_2 \in {}^V A$ conditions $P(d_1)\downarrow$ and $d_1 \subseteq d_2$ imply $P(d_2)\downarrow = P(d_1)$.

Any predicate $P : {}^V A \to \{T, F\}$ is uniquely determined by its truth and falsity domains:

$$T(P) = \{d \in {}^V A \mid P(d) = T\} \text{ and } F(P) = \{d \in {}^V A \mid P(d) = F\}.$$

A predicate $P : {}^V A \to \{T, F\}$ we call *irrefutable*, or partially true, if $F(P) = \varnothing$.

Special variable definedness predicates εz are used for description of properties of CNML of non-monotone predicates. The predicate εz explicitly indicates whether a variable $z \in V$ is defined (has a value). Such predicates εz are specified as follows:

$$T(\varepsilon z) = \{d \in {}^V A \mid z \notin asn(d)\};\ F(\varepsilon z) = \{d \in {}^V A \mid z \in asn(d)\}.$$

Predicates εz are non-monotone. Each $x \in V$ such that $x \neq z$ is unessential for εz.

The set C for pure first-order CNMS is denoted by basic logical compositions $\neg, \vee, \mathrm{R}^{\bar{v}}_{\bar{x}}, \exists x$ and basic modal compositions. Logical compositions $\rightarrow, \&, \leftrightarrow$ and $\forall x$ are derivative [4].

The compositions $\neg$ and $\vee$ we understand as strong Kleene negation and disjunction and define them by truth and falsity domains of the corresponding predicates:

$$T(\neg P) = F(P),\ F(\neg P) = T(P);$$

$$T(P \vee Q) = T(P) \cup T(Q),\ F(P \vee Q) = F(P) \cap F(Q).$$

Also we specify an unary renomination composition $\mathrm{R}^{\bar{v}}_{\bar{x}}$:

$$\mathrm{R}^{\bar{v}}_{\bar{x}}(f)(d) = f(\mathbf{r}^{\bar{v}}_{\bar{x}}(d)).$$

Language of pure first-order CNMS. Alphabet of a language includes a set of basic subject names (variables) V, a set Ps of predicate symbols (signature of the language), symbols of basic compositions $\neg, \vee, R^{\bar{v}}_{\bar{x}}, \exists x$, and a set of basic modal compositions Ms (modal signature).

The set Fm of formulas of the language is defined inductively:

FA) $Ps \subseteq Fm$; every predicate symbol $p \in Ps$ is an *atomic* formula;

FP) $\Phi, \Psi \in Fm \Rightarrow \neg\Phi, \vee\Phi\Psi \in Fm$;

FR) $\Phi \in Fm \Rightarrow R^{\bar{v}}_{\bar{x}}\Phi \in Fm$;

F$\exists$) $\Phi \in Fm \Rightarrow \exists x\Phi \in Fm$;

FM) $\Phi \in Fm$ and $\maltese \in Ms \Rightarrow \maltese\Phi \in Fm$.

We will call formulas modalised if they contain symbols from Ms.

For each $p \in Ps$, the set of its synthetically nonsignificant subject names is defined by a total mapping $\nu : Ps \rightarrow 2^V$, continued to formulas [4]: $\nu : Fm \rightarrow 2^V$. Also we have $\nu(\maltese\Phi) = \nu(\Phi)$.

The *type* of CNMS is specified by its modal signature Ms, similarity of relations in R for each $\maltese \in Ms$, and signature of synthetic nonsignificance $\sigma = (Ps, \nu)$.

Let us define an *interpretation mapping* of atomic formulas on states of the world $Im : Ps \times S \to Pr$. The condition $Im(p, \alpha) \in Pr_\alpha$ must hold: basic predicates are predicates of states.

We continue the interpretation Im to formulas $Im : Fm \times S \to Pr$ as follows:

IA) $Im(p, \alpha) = Im(p, \alpha)$ for each $p \in Ps$;

IP) $Im(\neg\Phi, \alpha) = \neg(Im(\Phi, \alpha))$;
$Im(\vee\Phi\Psi, \alpha) = \vee(Im(\Phi, \alpha), Im(\Psi, \alpha))$;

IR) $Im(R^{\bar{v}}_{\bar{x}}\Phi, \alpha) = \mathrm{R}^{\bar{v}}_{\bar{x}}\Phi(Im(\Phi, \alpha))$;

I$\exists$) $Im(\exists x\Phi, \alpha)(d) = \begin{cases} T, \text{ if } Im(\Phi,\alpha)(d\nabla x \mapsto a) = T \text{ for some } a \in A_\alpha, \\ F, \text{ if } Im(\Phi,\alpha)(d\nabla x \mapsto a) = F \text{ for all } a \in A_\alpha, \\ \text{else undefined.} \end{cases}$

IM) $Im(\maltese\Phi, \alpha)(d)$ is specified by values $Im(\Phi, \delta)(d)$ for certain states δ, such that α and δ are in some corresponding with $\maltese$ relations from R and may vary in form depending on the class of CNML.

Predicates which are the values of non-modalised formulas (they contain no modal symbols) are predicates of states.

Predicate $Im(\Phi, \alpha)$, which is the value of a formula Φ on a state α, we denote by Φ_α.

A formula Φ is *true on state* α (denoted as $\alpha \models \Phi$), if Φ_α is partially true (irrefutable) predicate.

Φ is *true for the CNMS* M (denoted as $M \models \Phi$), if Φ_α is partially true for each $\alpha \in S$.

A formula Φ is *everywhere (partially) true*, or irrefutable (denoted as $\models \Phi$), if $M \models \Phi$ holds for every CNMS of the same type.

2 Transitional modal systems

Transitional modal system (TMS) is a CNMS with a set R consisting of relations of the type $R \subseteq S \times S$ (i.e. transition relations).

Let us call a TMS *general*, if $R = \{\triangleright\}$ (there is a unique binary transition relation $\triangleright$) and it has one basic modal composition $\Box$ (necessarily).

Temporal transitional modal systems (TmMS): TMS with $R = \{\triangleright\}$ and basic modal compositions $\Box\uparrow$ (it will always be) and $\Box\downarrow$ (it was always).

For *multimodal* transitional modal systems (MMS), we have a number of basic modalities K_i, $i \in I$ with corresponding transition relations $\triangleright_i \in R$, therefore $R = \{\triangleright_i \mid i \in I\}$. Each K_i works as $\Box$, with respect to its own relation $\triangleright_i$, $i \in I$.

General TMS is actually a case of MMS.

Let us consider a language of pure first-order ***general*** TMS. Thus, $Ms = \{\Box\}$. To specify the set Fm of its formulas, we have got definitions FA, FP, FR, F$\exists$, and must clarify FM:

F$\Box$) if Φ is a formula, then $\Box\Phi$ is also a formula.

While defining the interpretation Im for formulas $\Box\Phi$, IM gets the following form:

I$\Box$) for each $\alpha \in S$ and $d \in {}^{V}A$:

$$Im(\Box\Phi, \alpha)(d) = \begin{cases} T, \text{ if } Im(\Phi, \delta)(d) = T \text{ for all } \delta \in S : \alpha \triangleright \delta, \\ F, \text{ if there is } \delta \in S : \alpha \triangleright \delta \text{ and } Im(\Phi, \delta)(d) = F, \\ \text{else undefined.} \end{cases}$$

In the case of non-existence for a given $\alpha \in S$ such a β that $\alpha \triangleright \beta$, we presume $Im(\Box\Phi, \alpha)(d)\uparrow$ (undefined value) for all $d \in {}^{V}A$.

Depending on restrictions on the relation $\triangleright$, various variants of general TMS can be defined. For example, let us consider reflexivity, transitivity, and symmetry. If $\triangleright$ is reflexive, we add an R-prefix to the name of the TMS, for transitivity we write T, and S means symmetry. Thus, we can get the following systems:
R-TMS, *T*-TMS, *S*-TMS, *RT*-TMS, *RS*-TMS, *TS*-TMS, *RTS*-TMS.

It should be noted that *R*-TMS is similar to *T*-system, *RS*-TMS is similar to *B*, *RT*-TMS to *S4*, and *RTS*-TMS to *S5*.

Now let us specify a language of pure first-order ***temporal*** TMS. We have $Ms = \{\Box\uparrow, \Box\downarrow\}$ and define FM as follows:

F$\Box\!\updownarrow$) if Φ is a formula, then $\Box\Phi\uparrow$ and $\Box\Phi\downarrow$ are also formulas.

The definition of IM for formulas of the types $\Box\Phi\uparrow$ and $\Box\Phi\downarrow$:

I$\Box\!\updownarrow$) for each $\alpha \in S$ and $d \in {}^{V}A$:

$$Im(\Box\uparrow\Phi, \alpha)(d) = \begin{cases} T, \text{ if } Im(\Phi, \delta)(d) = T \text{ for all } \delta \in S : \alpha \triangleright \delta, \\ F, \text{ if there is } \delta \in S : \alpha \triangleright \delta \text{ and } Im(\Phi, \delta)(d) = F, \\ \text{else undefined.} \end{cases}$$

$$Im(\Box\downarrow\Phi, \alpha)(d) = \begin{cases} T, \text{ if } Im(\Phi, \delta)(d) = T \text{ for all } \delta \in S : \delta \triangleright \alpha, \\ F, \text{ if there is } \delta \in S : \delta \triangleright \alpha \text{ and } Im(\Phi, \delta)(d) = F, \\ \text{else undefined.} \end{cases}$$

In the case of non-existence for a given $\alpha \in S$ such a β that $\alpha \triangleright \beta$, we presume $Im(\Box\uparrow\Phi, \alpha)(d)\uparrow$ (undefined value) for all $d \in {}^{V}A$.

In the case of non-existence for a given $\alpha \in S$ such a β that $\beta \triangleright \alpha$, we presume $Im(\Box\downarrow\Phi, \alpha)(d)\uparrow$ (undefined value) for all $d \in {}^{V}A$.

Depending on restrictions on the relation $\triangleright$, various variants of temporal TMS can be defined. Let us consider reflexivity, transitivity, and symmetry, thus, we can get the following systems:
R-TmMS, T-TmMS, S-TmMS, RT-TmMS, RS-TmMS, TS-TmMS, and RTS-TmMS.

Finally, let us specify a language of pure first-order ***multimodal*** TMS. We have $Ms = \{K_i \mid i \in I\}$ and need to define FM:

FK) if Φ is a formula and $K_i \in Ms$ then $K_i\Phi$ is also a formula.

The definition of IM for formulas of the types $K_i\Phi$:

IK) for each $\alpha \in S$ and $d \in {}^{V}A$:

$$Im(K_i\Phi, \alpha)(d) = \begin{cases} T, \text{ if } Im(\Phi, \delta)(d) = T \text{ for all } \delta \in S : \alpha \triangleright_i \delta, \\ F, \text{ if there is } \delta \in S : \alpha \triangleright_i \delta \text{ and } Im(\Phi, \delta)(d) = F, \\ \text{else undefined.} \end{cases}$$

In the case of non-existence for a given $\alpha \in S$ such a β that $\alpha \triangleright_i \beta$, we presume $Im(K_i\Phi, \alpha)(d)\uparrow$ (undefined value) for all $d \in {}^V A$.

Let us call a MMS *epictemic* (EMS) if the set $R = \{\triangleright_i\}$ is finite.

We can obtain the following variants of epictemic MMS:
R-EMS, T-EMS, S-EMS, RT-EMS, RS-EMS, TS-EMS, RTS-EMS.

Traditional epistemic systems can be considered within the scope of EMS. For instance, R-EMS, T-EMS and RTS-EMS are the generalisations of $T_{(n)}$, $S4_{(n)}$ and $S5_{(n)}$ correspondingly.

At the same time, there can exist MMS of mixed types when, for example, $\triangleright_1$ is transitive, $\triangleright_2$ is transitive and reflexive, $\triangleright_3$ is symmetric etc.

3 Properties of transitional modal systems

For TML of non-monotone quasiary predicates (the equitone (monotone) case was studied in [4-7]) we distinguish global predicates and predicates of states. Predicates which are the values of non-modalised formulas are predicates of states; they are specified as follows: $\Phi_\delta(d) = \Phi_\delta(d_\delta)$ for an arbitrary $d \in {}^V A$, where d_δ is a nominative set $[v \mapsto a \in d \,|\, a \in A_\delta]$. Informally, it means that predicates of states "perceive" only such components $v \mapsto a$ that $a \in A_\delta$. Global predicates are the values of modalised formulas.

Modal compositions of TMS can be carried over renominations:

Theorem 1. $R^{\bar{v}}_{\bar{x}} \maltese \Phi(d) = \maltese R^{\bar{v}}_{\bar{x}} \Phi(d)$ *for arbitrary* $\maltese \in Ms$, Φ, $d \in {}^V A$.

Proof is similar to the equitone (monotone) case of TML (see [5]).

Corollary 1. *Formulas of the type* $R^{\bar{v}}_{\bar{x}} \maltese \Phi(d) \leftrightarrow \maltese R^{\bar{v}}_{\bar{x}} \Phi(d)$ *are everywhere true* ($\maltese \in Ms$).

Let us consider the interaction between modal compositions and quantifiers in TMS.

In the case of TML of non-monotone (non-equitone) predicates, formulas of the types $\maltese \forall x \Phi \to \forall x \maltese \Phi$ and $\exists x \maltese \Phi \to \maltese \exists x \Phi$ are not true.

Example 1. *Formulas* $\Box \forall x\, \Phi \to \forall x \Box \Phi$ *and* $\exists x \Box \Phi \to \Box \exists x\, \Phi$ *are not true.*

Proof. Let us specify a general TMS in which the formula $\Box\forall x\Phi \to \forall x\Box\Phi$ is refuted. Assume $S=\{\alpha,\beta\}$, $R=\{\alpha \triangleright \beta\}$, $A_\alpha=\{a\}$, $A_\beta=\{b\}$. Let us take $p \in Ps$ for which all the names are unessential but x. Let $p_\alpha(\varnothing) = F$, $p_\beta(\varnothing) = F$, $p_\beta([x \mapsto b]) = T$. Then $(\forall x\, p)_\beta(\varnothing) = T$ and $(\Box\forall x\, p)_\alpha(\varnothing) = T$. From $[x \mapsto a]_\beta = \varnothing$ we have $p_\beta([x \mapsto a]_\beta) = p_\beta(\varnothing) = F$ and $(\Box p)_\alpha([x \mapsto a]) = F$. Then $(\forall x\Box p)_\alpha(\varnothing) = F$. Thus $(\Box\forall x\, p \to \forall x\Box p)_\alpha(\varnothing) = F$ and $\alpha \not\models \Box\forall x\, p \to \forall x\Box p$. Notice that the predicate p_β is non-equitone.
A general TMS in which $\exists x\Box\Phi \to \Box\exists x\Phi$ is refuted can be specified by analogy. □

At the same time, formulas of the types $✠\forall x\Phi \to \forall x✠\Phi$ and $\exists x✠\Phi \to ✠\exists x\Phi$ are everywhere true in the case of TML of equitone predicates ([7]).

However, formulas of the types $\forall x✠\Phi \to ✠\forall x\Phi$ and $✠\exists x\Phi \to \exists x✠\Phi$ are not true ([5]) already in the equitone case of TML. In particular, the formulas $\forall x\Box\Phi \to \Box\forall x\Phi$ and $\Box\exists x\Phi \to \exists x\Box\Phi$ are not true.

$\forall x\Box\Phi \to \Box\forall x\Phi$ is known as the Barcan formula, and $\Box\forall x\Phi \to \forall x\Box\Phi$ is the converse Barcan formula. As we showed in Example 1, the converse Barcan formula is not true, but this formula is true for every general TMS of equitone predicates.

4 Logical consequence for sets of formulas, specified with states

Let us denote a formula specified with a state as Φ^α. Here Φ is a formula of a language, $\alpha \in S$ (S is a set of names of states of the universe) – its specification; the specification indicates a state in which Φ is considered.

Let M is TMS with a set of states S, Γ is a set of formulas, specified with states such that the states compose a set $\mathbf{S}$.

A set Γ is *agreed* with TMS M, if an injection $\mathbf{S}$ to S is specified.

Let Γ and Δ are sets of formulas, specified with states.

Δ is a *consequence* of Γ in the agreed with them CNMS M (denoted $\Gamma_M \models \Delta$), if for all $d \in {}^VA$: $\Phi_\alpha(d) = T$ for all $\Phi^\alpha \in \Gamma$ implies the

impossibility that $\Psi_\beta(d) = F$ for all $\Psi^\beta \in \Delta$.

Δ is a *logical consequence* of Γ (denoted $\Gamma \models \Delta$) if $\Gamma_M \models \Delta$ holds for all CNMS M of the same type. Thus, $\Gamma \not\models \Delta \Leftrightarrow$ there is a CNMS M agreed with Γ and Δ, and $d \in {}^V A$ such that $\Phi_\alpha(d) = T$ for all $\Phi^\alpha \in \Gamma$ and $\Psi_\beta(d) = F$ for all $\Psi^\beta \in \Delta$.

Let us consider properties of a consequence relation for sets of formulas, specified with states in the given CNMS.

The non-modal properties are identical to the corresponding properties for logics of quasiary predicates (see [3,4]). As an example let us list basic properties of propositional level and of quantifier elimination.

C) If $\Gamma \cap \Delta \neq \varnothing$ then $\Gamma_M \models \Delta$. This property guarantees an existence of the consequence;

$\neg_L$) $\neg\Phi^\alpha, \Gamma_M \models \Delta \Leftrightarrow \Gamma_M \models \Delta, \Phi^\alpha$;

$\neg_R$) $\Gamma_M \models \Delta, \neg\Phi^\alpha \Leftrightarrow \Phi^\alpha, \Gamma_M \models \Delta$;

$\vee_L$) $\vee\Phi\Psi^\alpha, \Gamma_M \models \Delta \Leftrightarrow \Phi^\alpha, \Gamma_M \models \Delta$ and $\Psi^\alpha, \Gamma_M \models \Delta$;

$\vee_R$) $\Gamma_M \models \Delta, \vee\Phi\Psi^\alpha \Leftrightarrow \Gamma_M \models \Delta, \Phi^\alpha, \Psi^\alpha$.

Basic properties of quantifier elimination:

$\exists_L$) $\exists x\Phi^\alpha, \Gamma_M \models \Delta \Leftrightarrow \mathrm{R}^x_z\Phi^\alpha, \Gamma_M \models \Delta, \varepsilon z^\alpha$, here $z \in V_T$ and $z \notin nm(\Gamma, \Delta, \exists x\Phi)$;

$\exists \mathrm{v}_R$) $\Gamma_M \models \Delta, \exists x\Phi^\alpha, \varepsilon y^\alpha \Leftrightarrow \Gamma_M \models \Delta, \exists x\Phi^\alpha, \mathrm{R}^x_y\Phi^\alpha, \varepsilon y^\alpha$.

When eliminating quantifiers we use the following properties (here V_T denotes a set of globally unessential names [4]):

εd) $\Gamma_M \models \Delta \Leftrightarrow \varepsilon y^\alpha, \Gamma_M \models \Delta$ and $\Gamma_M \models \Delta, \varepsilon y^\alpha$;

εv) $\Gamma \models \Delta \Leftrightarrow \Gamma \models \Delta, \varepsilon z^\alpha$, where $z \in V_T$ and $z \notin nm(\Gamma, \Delta)$.

Now let us consider properties concerned with modalities in the case of general TMS.

Carring modalities over renominations:

R$\Box_{\rm L}$) $\Gamma, {\rm R}_{\bar{x}}^{\bar{v}}\Box\Phi^{\alpha}\,{}_{M}\models\Delta \Leftrightarrow \Gamma, \Box{\rm R}_{\bar{x}}^{\bar{v}}\Phi^{\alpha}\,{}_{M}\models\Delta$;

R$\Box_{\rm R}$) $\Gamma_M \models \Delta, {\rm R}_{\bar{x}}^{\bar{v}}\Box\Phi^{\alpha} \Leftrightarrow \Gamma_M \models \Delta, \Box{\rm R}_{\bar{x}}^{\bar{v}}\Phi^{\alpha}$.

Elimination of modalities:

$\Box_{\rm L}$) $\Box\Phi^{\alpha}, \Gamma_M \models \Delta \Leftrightarrow \{\Phi^{\beta} \mid \alpha \triangleright \beta\} \cup \Gamma_M \models \Delta$;

$\Box_{\rm R}$) $\Gamma_M \models \Delta, \Box\Phi^{\alpha} \Leftrightarrow \Gamma_M \models \Delta, \Phi^{\beta}$ for all $\beta \in S$ such that $\alpha \triangleright \beta$.

The properties of consequence for sets of formulas, specified with states, in the given CNMS induce the corresponding properties of *logical* consequence for sets of the such formulas. The latter properties can be the semantic base for construction of sequent calculi for various classes of TML.

The property C gives us the closure condition of a sequent; closed sequents are *axioms* of sequent calculi. All the other properties can be transformed into sequent forms – the *inference rules* of sequent calculi.

In sequent calculi, derivations have a form of a tree with sequents as nodes. Sequent tree is *closed*, if all its leafs are closed sequents. A sequent Σ is *derivable*, or *has a derivation*, if there exists a closed sequent tree with a root Σ; such tree is called a *derivation* of a sequent Σ.

By building a sequent tree we are in fact checking the *absence* of a logical consequence. Hence, the sequent forms must preserve $\not\models$ during the decomposition of a formula, and $\models$ during the composition of a formula from components. So properties of ${}_{M}\not\models$, dual to the ones of ${}_{M}\models$, have to be considered.

We specify dual properties for $\vee_L$ and $\Box_{\rm R}$ as follows:

${}_{L}\vee$) $\vee\Phi\Psi^{\alpha}, \Gamma_M \not\models \Delta \Leftrightarrow \Phi^{\alpha}, \Gamma_M \not\models \Delta$ or $\Psi^{\alpha}, \Gamma_M \not\models \Delta$;

${}_{\rm R}\Box$) $\Gamma_M \not\models \Delta, \Box\Phi^{\alpha} \Leftrightarrow \Gamma_M \not\models \Delta, \Phi^{\beta}$ for some $\beta \in S$ such that $\alpha \triangleright \beta$.

As $P \Leftrightarrow Q$ is equivalent to $\neg P \Leftrightarrow \neg Q$, the rest of the properties for ${}_{M}\not\models$ can be obtained simply by replacing ${}_{M}\models$ by ${}_{M}\not\models$, like here:

${}_{\rm L}\Box$) $\Box\Phi^{\alpha}, \Gamma_M \not\models \Delta \Leftrightarrow \{\Phi^{\beta} \mid \alpha \triangleright \beta\} \cup \Gamma_M \not\models \Delta$.

Additional conditions imposed on transition relations modify properties of modalities elimination. Let us consider the dual properties of elimination of modalities for general TMS and a reflexive, transitive and symmetric $\triangleright$ (all possible combinations).

1. $\triangleright$ *is reflexive:* $\alpha \triangleright \alpha$, so for ${}_{\mathrm{L}}\Box$ we have $\Phi^{\alpha} \in \{\Phi^{\beta} \mid \alpha \triangleright \beta\}$. The property ${}_{\mathrm{R}}\Box$ is the same as in general case.

2. $\triangleright$ *is symmetric:* $\alpha \triangleright \beta \Leftrightarrow \beta \triangleright \alpha$, so ${}_{\mathrm{L}}\Box$ and ${}_{\mathrm{R}}\Box$ have forms

${}_{\mathrm{L}}\Box$S) $\Box\Phi^{\alpha}, \Gamma_M \not\models \Delta \Leftrightarrow \{\Phi^{\beta} \mid \alpha \triangleright \beta \text{ or } \beta \triangleright \alpha\} \cup \Gamma_M \not\models \Delta$;

${}_{\mathrm{R}}\Box$S) $\Gamma_M \not\models \Delta, \Box\Phi^{\alpha} \Leftrightarrow \Gamma_M \not\models \Delta, \Phi^{\beta}$ for some $\beta \in S$ such that $\alpha \triangleright \beta$ or $\beta \triangleright \alpha$.

3. $\triangleright$ *is reflexive and symmetric.* We have ${}_{\mathrm{R}}\Box$S and

${}_{\mathrm{L}}\Box$RS) $\Box\Phi^{\alpha}, \Gamma_M \not\models \Delta \Leftrightarrow \{\Phi^{\beta} \mid \alpha \triangleright \beta \text{ or } \beta \triangleright \alpha\} \cup \Gamma_M \not\models \Delta$ and $\Phi^{\alpha} \in \{\Phi^{\beta} \mid \alpha \triangleright \beta \text{ or } \beta \triangleright \alpha\}$.

4. $\triangleright$ *is transitive.* We have ${}_{\mathrm{R}}\Box$ and

${}_{\mathrm{L}}\Box$T) $\Box\Phi^{\alpha}, \Gamma_M \not\models \Delta \Leftrightarrow \{\Phi^{\beta} \mid \alpha \triangleright \beta\} \cup \{\Box\Phi^{\beta} \mid \alpha \triangleright \beta\} \cup \Gamma_M \not\models \Delta$. Note that $\{\Box\Phi^{\beta} \mid \alpha \triangleright \beta\}$ is needed because $\triangleright$ is transitive.

5. $\triangleright$ *is transitive and reflexive.* We have ${}_{\mathrm{R}}\Box$ and ${}_{\mathrm{L}}\Box$T. Because of reflexivity, we need $\Phi^{\alpha} \in \{\Phi^{\beta} \mid \alpha \triangleright \beta\}$ for ${}_{\mathrm{L}}\Box$T.

6. $\triangleright$ *is transitive and symmetric.* We have ${}_{\mathrm{R}}\Box$S and

${}_{\mathrm{L}}\Box$TS) $\Box\Phi^{\alpha}, \Gamma_M \not\models \Delta \Leftrightarrow \{\Phi^{\beta} \mid \alpha \triangleright \beta \text{ or } \beta \triangleright \alpha\} \cup \{\Box\Phi^{\beta} \mid \alpha \triangleright \beta \text{ or } \beta \triangleright \alpha\} \cup \Gamma_M \not\models \Delta$.

7. $\triangleright$ *is transitive, reflexive and symmetric.* We have ${}_{\mathrm{R}}\Box$S and ${}_{\mathrm{L}}\Box$TS. We need $\Phi^{\alpha} \in \{\Phi^{\beta} \mid \alpha \triangleright \beta \text{ or } \beta \triangleright \alpha\}$ for ${}_{\mathrm{L}}\Box$TS because of reflexivity.

Basing on the considered properties of logical consequence relation for sets of formulas, specified with states, the corresponding sequent calculi for various classes of TML can be constructed.

5 Conclusion

In this paper program-oriented logical formalisms with modalities were studied. We introduced pure first-order transitional modal logics of partial predicates without monotonicity (equitonicity) restriction. For such logics, the languages and semantic models were defined and their semantic properties were considered; the difference between MTL of monotone and non-monotone predicates was demonstrated, as an example, the converse Barcan formula is not valid in the case of non-monotone predicates, however, this formula is valid in the case of monotone predicates [5]. We investigated properties of logical consequence relations for sets of formulas, specified with states, for various classes of logics, which can be the base for sequent calculi construction in the future.

References

[1] S. Abramsky, D. Gabbay, T.S.E. Maibaum. *Handbook of Logic in Computer Science: in 5 vol.* Clarendon Press, Oxford, 1994-2000.

[2] D. Bjørner, M.C. Henson. *Logics of Specification Languages.* Springer, Heidelberg, 2008.

[3] M. Nikitchenko, S. Shkilniak. *Semantic Properties and Sequent Calculi of Pure First-order Composition Nominative Logics.* Information Theories and Applications, vol. 20, no 4, pp. 379–390, Sofia, Bulgaria, 2013. (in Russian)

[4] M. Nikitchenko, S. Shkilniak. *Applied Logic.* Publishing house of Taras Shevchenko National University of Kyiv, Kyiv, 2013. (in Ukrainian)

[5] O. Shkilniak. *Semantic Properties of Composition Nominative Modal Logics.* Problems in Programming, no 4, 2009, pp. 11–23. (in Ukrainian)

[6] O. Shkilniak. *Semantic Models and Sequent Calculi of Transitional Modal Logics.* Computer Mathematics, no 1, 2013, pp. 141–150. (in Ukrainian)

[7] O. Shkilniak. *Modal Logics of Partial Predicates and Sequent Systems of Logical Reasoning in Them.* Information Theories and Applications, vol. 20, no 4, pp. 367–378, Sofia, Bulgaria, 2013. (in Russian)

Oksana Shkilniak

Received July 12, 2015

Taras Shevchenko National University of Kyiv,
Faculty of Cybernetics, Department of Information Systems
64/13, Volodymyrska Street, City of Kyiv, Ukraine, 01601
Phone: +380442590511
E–mail: me.oksana@gmail.com

Part 6

Formal languages and automata

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Communicative automata based programming. Society Framework*

Andrei Micu, Adrian Iftene

Abstract

One of the aims of this paper is to present a new programming paradigm based on the new paradigms intensively used in IT industry. Implementation of these techniques can improve the quality of code through modularization, not only in terms of entities used by a program, but also in terms of states in which they pass. Another aspect followed in this paper takes into account that in the development of software applications, the transition from the design to the source code is a very expensive step in terms of effort and time spent. Diagrams can hide very important details for simplicity of understanding, which can lead to incorrect or incomplete implementations. To improve this process communicative automaton based programming comes with an intermediate step. We will see how it goes after creating modeling diagrams to communicative automata and then to writing code for each of them. We show how the transition from one step to another is much easier and intuitive.

Keywords: Communicative Automata, XML Automata, Automata transfer, Distributed systems, Traveling code.

1 Introduction

The last decades of evolution for computing machines brought a significant increase in computing power and their diversity. The rise of parallel computing, the important foundations of modern computers, has revolutionized the world of software and hardware making it possible to create artificial intelligent systems. Whether it is a desktop,

* This work was supported by project 2 CHIST-ERA/01.10.2012

mobile phone, mainframe, or any other computing system, it is able to simultaneously perform a number of tasks that sometimes depend on each other. Their synchronization is essential in most cases and it depends on the states in which the processes or threads are at a time. Synchronization is not easy to achieve if the source code is not structured in terms of states. Sometimes it happens that a seemingly stable code in terms of errors to work as expected for successive runs with the same input data, but at a certain running (with the same input data) to give a wrong result.

Automata have been used since before the beginning of modern computers to solve mathematical problems. Nowadays they have applications in many of the components of a software product such as lexical analyzers, parsers which use regular expressions or network communication protocols [1]. In 2003 Russian scientist Anatoly Shalyto published an article about automata based programming [2]. This paper presents a new way of programming mechanisms for simulation of states, transitions and input/output operations. In designing of large applications state charts and activity diagrams can be used, and they are very similar to automata. The problem arises when you have to translate these diagrams into source code. Shalyto senses this transposition and connects his theory with the association between diagram elements and automata elements.

Communicative automata based programming has elements from the object-oriented version of the Russian researcher and, additionally, it solves the problems mentioned above. It proposes an improved model of the application, dividing the tasks of the control automata in the object-oriented model to several independent machines that communicate with each other. Every communicative automata is self-contained and do not share information, the only way to exchange data is the transmission of messages. So the code of applications benefits from high cohesion without sacrificing coupling and it can be reused easily.

The main strength in the technology based on communicative automata is that the application can be easily distributed across multiple computing machines. Each system has its suite of communicative automata, which communicates with the rest of the automata by the

same type of messages; in this case the communication channel is the network. Moreover, these systems can switch automata between them, which do not depend on a particular machine, providing task balancing. This is useful particularly in the client-server model, where the client is a computing machine with low capacities, such as a mobile phone. In this paper we will see how we created the premises for the communicative automata based programming paradigm, which is based on object-oriented programming concepts. This paradigm intensively uses the concept of automaton; code structure is given by the states and transitions. Novelty to classic automata based programming is to treat automata as atomic elements at application-level and the introduction of using transmission of messages between automata. Society Framework implements the basic elements and concepts described by communicative automata based programming. The framework allows creation of native and XML automata, the XML ones having an important advantage because they can be serialized locally and deserialized on another machine at runtime.

2 Communicative automata based programming

2.1 Automata based programming (classical version)

For the first time in software engineering, Shalyto describes an application model composed only from automata. Its technology uses intensive enumerations and switch-case instructions, so that is also called "Switch Technology" [2]. In this solution, although the code is easy to understand, there are big problems when the number of states increases. This is because the program code increases with each added state and transition. The solution appears later in the paradigm "object-oriented programming based on states" [2]. It combines the advantages of automata for easier understanding of the program behavior and advantages of object-oriented programming for easier understanding of the structure. The new technique shifts from "switch-case" instructions to classes and objects to describe automata, thus avoiding

nested switches. Shalyto takes in discussion the existence of several automata in one application. Their management is difficult when you have to synchronize certain transitions or when performing operations of reading/writing from the same memory location. Therefore the notion of "space of states" [2] arises, a set of conditions designed to control objects. Knowing which states control objects, we can program automata to have synchronous access to memory. Moreover, if we consider one object for each automaton, the space of states may act as a supervisor for the other automata. Thus, the application model begins to look like client-server model, as in Figure 1.

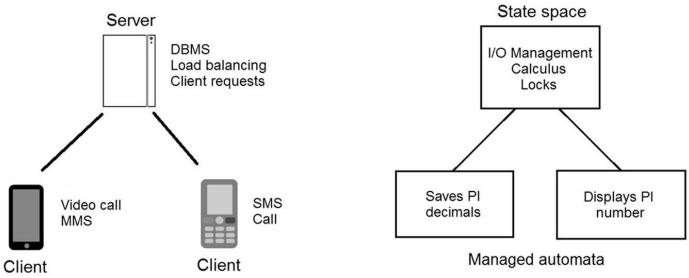


Figure 1. Comparison between client-server model and Shalyto model

Another novelty in this technique is the capacity of system to respond to events, a necessary feature for communication between state space and the other automata. So, not only I/O operations can change the state of automata, but also other automata by generating its own events.

2.2 Communicative automata based programming

Programming based on interconnected automata expands object-oriented programming, inheriting all its elements and rules. On top of them there is the added notion of communicative automaton, with new rules related to its functionality. The major differences between the automata from object-oriented version and communicative automata

are their atomic characteristic and the communication based on messages. In what follows, an automaton is a finite automaton with epsilon-transitions, without isolated states, with one or more final states and a single initial state [3].

Communicative automata bring new elements compared to the classical automata, leading to changing application architecture. The main elements are: (1) **State** – a series of instructions viewed as an atomic part. It contains code that performs the actual work of the automaton; (2) **Transition** – a series of instructions viewed as an atomic part. In contrast with states, transitions contain only the code needed to determine the next state where it will pass; (3) **Message** – an entity that contains data transmitted by an automaton to another automaton or by a code that is not part of an automaton; (4) **List of messages** – a comprehensive list of received messages by automaton.

A communicative automaton may contain, in addition to the base elements, other resources such as variables, operating system resources or references/pointers to other automata. In most cases, when a system runs more communicative automata, it is desirable to execute their code in parallel. The general solution in modern systems is to run the code for each automaton on a separate thread. Some programming languages such as JavaScript before HTML5 [4, 5], do not support working with multiple threads. If the parallel execution is not possible, the programmer will have to use an own method of allocation and arbitration of automata to the processor. This approach has an important advantage in that the programmer can choose the convenient moments when to deallocate an automaton to ensure a consistent state at each step of the execution. This approach has a disadvantage, though. At any point in execution it cannot run more than one automaton, so others have to wait.

Each automaton must provide ways to add messages to it message list. The problem occurs when an automaton should reference another automaton, to which it must send a message. A naive way to solve this problem is to keep a reference/pointer to every possible destination, which is set by the function that creates it. Such practices, however, are extremely hard to maintain since it requires changing the code

of the function that instantiates automata each time when we add a new type of automaton. A better approach is to mediate communication with an object that keeps track of automata. This object, called "router", has an implementation similar with the Observer pattern. Automata must register using a unique identifier to this mediator and must be able to handle messages from any source, automaton or not. The router methods can send messages based on recipient identifier, thus obtaining a total decoupling of automata in the system. When the router handles messages sent through the network, security problems may appear which must be taken into account. A security system must provide a separation between user automata to avoid situations in which an automaton sends a compromising message to another automaton.

3 Society framework

3.1 Architecture

Society Framework is a project developed to demonstrate the advantages of communicative automata based programming. Its source code and examples are publicly available on https://code.google.com/p/society-framework/. Its target is to facilitate the development of communicative automata based applications and to minimize the errors that can happen in such an application. There are two framework implementations, one written in Java language and one written in C# language, the reason being the demonstration of its interoperability.

The three major modules of this framework are the following: (1) **Base module for communicative automata** – contains interfaces, abstract classes and completely implemented classes to create a native automaton; (2) **XML communicative automata module** – contains interfaces, abstract classes and completely implemented classes to create an XML automaton; (3) **Communication module** – contains classes with role in automata communication, both locally and through the network, on different applications.

In Society the communicative automata are divided in two large categories: native automata, with Java or C# code based on the frame-

work implementation and XML automata, for which the code is written using an extension of XML. The main reason is the fact that, unlike the native automata, the XML ones can be serialized, sent through the network to another application and re-instantiated on that machine. The XML code inside the serialization is transformed in one more object trees representing the instruction that must be executed in the states and the transitions. For this reason we cannot say that the framework compiles the code and neither that it interprets it. It constructs its own code using objects corresponding to the instructions described by the XML.

XML automata can use only a restricted set of instructions, on which the framework can construct XML specifications. The reason is the fact that XML automata are not intended for complex or intense processing due to the great overhead compared to the native ones. Their utility is the fact that they can communicate with the native ones through messages, the latter fulfilling the tasks described in the messages much more efficiently. The alternative usage of the two types enables optimizing the application processing, an example being described in Figure 2.

The efficiency relies on the fact that the only data sent between the server and the client is the automaton serializations. Thus, the transfer of data between server and client is minimized and it is replaced by service calls, the result being a constant number of connections to the server for executing a task. Fortunately an XML automaton plays the role of manager for the native automata in the system and they don't require much code, so traveling to another machine through network doesn't imply a lot of data transfer. Another major advantage in this approach is the reduced overhead of the native automata because, as we previously mentioned, the native automata are written directly in the language/platform of the framework implementation. The fact that a native automaton can use any instruction of this type raises security problems regarding the actions permitted to XML automata. XML automata can only execute instructions that don't compromise security, the only way of communicating with the machine it runs on being the message sending to other automata.

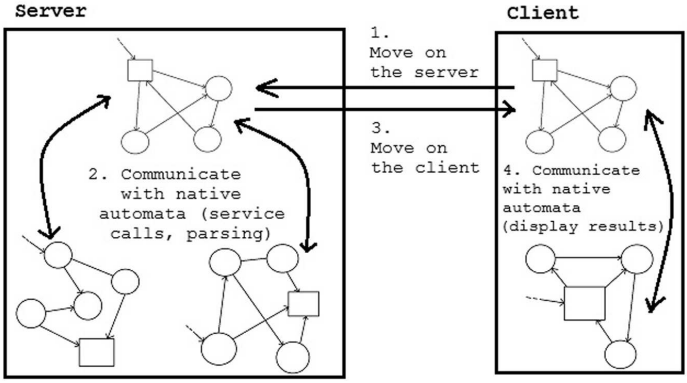


Figure 2. Example of optimization using a native automaton and an XML automaton

3.2 Base module for communicative automata

The base module contains the base class for all the automata created in a system (BaseAutomaton) and classes for automaton components. In both framework implementations these are: (1) **State** – the base class for the states in an automaton; (2) **Transition** – the base class for the transitions in an automaton; (3) **TransitionGroup** – class that contains transitions and acts like a normal transition; (4) **MessageList** – the class that implements the queue of messages received by an automaton.

The MessageList class implements a special type of queue with synchronized add and remove methods (see Figure 3). The add operation is synchronized to ensure consistency to the list while more than one execution threads add messages simultaneously. The remove operation is synchronized to block the thread that calls it when there are no messages in the queue until another thread places a message in it. The message queue is encapsulated in the automaton, the only permitted operation being the message addition by using the Add method at automaton.

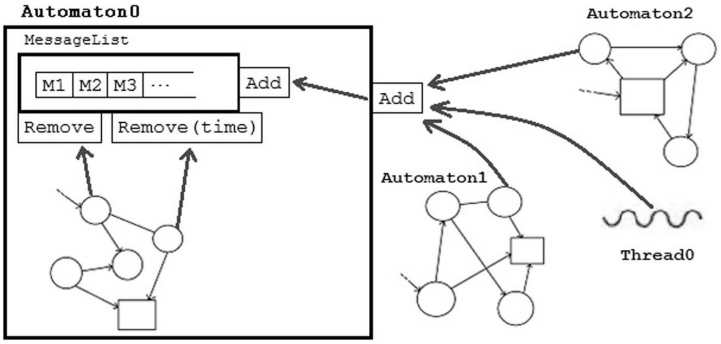


Figure 3. Interaction with the message list (queue)

Each state contains a transition or a transition group which indicates the next state and is kept in an indexed list (HashMap in Java and Dictionary in C#) with string identifiers. Society's communicative automata can run both on the current thread, by calling the run (or Run in C#) method, and on a newly created thread by calling start (or Start in C#).

Another important class in the base module is the SocietyManager which manages the automata inventory from the current application and the XML automata transfers. This can be extended to implement the saving and loading methods for the automata. For automata transfer SocietyManager runs a special native automaton called AutomataTransferAutomaton responsible with managing the connections and sending/receiving automata.

3.3 XML communicative automata module

XML communicative automata are an extension of the native ones, their base class being BaseAutomaton. The XML automaton name is given by the serialization and deserialization of this type, which is achieved by using the Society XML, an extension of XML.

The serialization of XML automata contains 6 major elements: (1) **Automaton name** – also named identifier, it is the character sequence attribute of the < *automaton* > element with role in a possible subscribe of the automaton at the message router; (2) **Current state name** – current state identifier, also stored as a character sequence; (3) **Current message** – the serialization of the last message pulled from the message queue; (4) **Message list** – the serialization of the automaton's message queue; (5) **Variables** – the list of variables and their values; (6) **States** – the list of states in the automaton and their code.

The current state name must match the name of a state in the state list. If this rule is violated then the automaton cannot start. Also, if the name which was provided for the automaton serialization differs from the name it had in the system before, when the automaton is reinstantiated all entities which send messages to that automaton must be aware of the change and send message for the new name.

Variables are key-value pairs. In the Society framework automata work with 5 data types: Boolean, Integer, Double, String and Map. Map type can represent vectors with any number of dimensions; the only limit is the machine memory. To ensure this property, the framework uses a series of indexed lists (the type of list depends by framework implementation). Thanks to this flexibility we can create vectors that contain other vectors, any number of times and in any combination, the result being as much dimensions as the memory can hold.

The state serialization contains the state identifier (name attribute), the executed code (< *code* > element) and the transition (< *transition* > element) or the transition group (< *transition_group* > element). A transition group contains any number of transitions and a < *code* > element which contains the instructions that manage the returned values for each transition. From the moment when the automaton is started, the initial state code is executed indicated by the value in the *current_state* attribute. Then the transition code from that state is executed (or the transition group code if it's a group of transitions) and the next state name is obtained. The process continues until the automaton is stopped, either from a state code, or from

an execution thread outside it.

It should be noted the fact that inside a transition group each transition must have a name (specified in the name attribute), so that their code can be called from the $< code >$ element of the group. If a state has just a transition, and not a transition group, then that transition doesn't have to specify a name. The instructions represent the imperative part of Society XML and there are two types of them: (1) **Simple instructions** – instructions that don't contain other instructions inside them (empty elements); (2) **Compound instructions** – instructions that contain other instructions inside them (non-empty elements).

Simple instructions are the base for the code executed in the states and transitions. These are represented by XML elements without content, the only parameters being their attributes. The following simple instructions can be used in Society XML: *get_next_message*, *send_message*, *execute*, *continue*, *break* and *return*. Compound instructions are usually loops (*while*, $do - while$, for), but they can be other types, like the $if - else$ instruction or the $switch - case$. Their behavior is identical with the one in the framework's implementation language, with small differences to enable much more flexibility by using the expressions. The following compound instructions can be used in Society XML: *while*, *do_while*, for, if, *else*, *switch*, *case* and $default$.

Unlike the other elements, where the declaring order does not drive the code behavior, instructions must be written in the exact order in which they must be executed. Regarding the calculability of Society XML, it contains enough instructions to be Turing-complete: at least one assignation operation, one conditional operation and one jump instruction. This means that XML communicative automata can solve any problem which can be transformed in an algorithm. The input and the output are ensured by the router and the message queue inside the automaton. Native automata must provide communication methods with the user for the XML ones because the language of the latter does not allow native calls for reading, writing or displaying data. The advantage is the fact that XML automata don't have any security issues so long as the native ones verify and control the requests. The ex-

pressions have an important role in Society XML because they provide values for the attributes in the instructions presented earlier.

XML communicative automata deserialization

The BaseAutomaton class incorporates methods to serialize and deserialize automata. For parsing the XML data the Society framework uses SAXParser in Java and XMLReader in C#. For constructing the objects that compose the XML automaton functionality the framework uses a special automaton called DeserializationAutomaton. For each beginning and ending tag the parser sends a message to the deserialization automaton containing the corresponding data (tag name, attributes, and tag type). According to the state and message data the automaton will create the objects and perform transitions.

XML communicative automata serialization

XML automata serialization is a much simpler process thanks to the tree structure of the instructions inside them. The serialization process implies the construction of the XML based on the objects inside the automaton. To achieve this it is necessary to inspect the variables, the message list, the special members (automaton name, current message, etc.) and a single BFS traversal of object trees in the states, transitions and transition groups.

3.4 The communication module

The communication module includes the classes responsible with sending and routing the messages: (1) **Message** – the class which represents a message; (2) **MessageRouter** – the class responsible with message transfers.

The Message class contains two fields (or properties in C#): $from$ and *data*. The $from$ field holds the identifier with which the sender automaton has subscribed to the router or any other name if the message was not sent by an automaton. The *data* field is a reference to the sent object and it can be of any type.

To send messages to an automaton outside the application the router uses an automaton which is responsible with the message transfer through the network called NetworkMessagingAutomaton. This

looks similar to the DeserializationAutomaton inside the SocietyManager class, the only difference being the type of sent information.

4 Comparisons

4.1 Loose code vs. native communicative automata

Loose code means any object-oriented code written without the constraints of the communicative automaton based programming paradigm. By applying these constraints to loose code the native communicative automatons can be obtained, the performance difference being minimal.

To create a native automaton the following steps must be followed: (1) **Extending the BaseAutomaton or Automaton classes** – if the automaton state doesn't have to be persisted then the Automaton class is used, otherwise the BaseAutomaton class is extended and the serialization/deserialization methods are implemented; (2) **Adding the member variables** – necessary for the automaton functionality; (3) **Creating the nested classes** – corresponding to the states, transitions and transition groups; (4) **Instantiating and adding the previously created classes at the current automaton** – these steps can be made in the automaton constructor.

The code executed in the stateCode and transitionCode methods by the automaton is the imperative (procedural) code corresponding to the language of the framework implementation. The run method (or Run in C#), which was previously mentioned in this article, is responsible with the correct execution of the automaton regarding the order in which the states, the transitions and the transition groups are executed. The management instructions in this method have O(1) complexity, except the operation that searches a state in the state set. After a transition returns an identifier, in the run method, a search for the state corresponding to that identifier is attempted. This implies a get operation in a HashMap (Java) or Dictionary (C#) with a complexity in the worst case scenario of O(n) [7, 8], where n is the length of the identifier hash. While the automata are designed, the programmers

and architects must decide how to split the automaton tasks and what are their states and transitions.

4.2 Native communicative automata vs. XML communicative automata

The XML communicative automata, as mentioned in the previous chapters, are an extension of the native communicative automata. Their states, transitions and transition groups are constructed in a certain manner to ensure they can be serialized and deserialized using the Society XML language. The code inside the XML automaton is composed of an object tree and the objects are implementing the Instruction interface. Executing its code means calling the code method of the root object which will trigger directly or indirectly the call of code in the other objects from the tree.

The overhead compared to the native ones is visibly greater because each instruction implies a method call at the level of implementation. Starting from the moment the automaton tries to assign a value to a Map object at an inexistent level, the algorithm creates the necessary levels based on the indexes from the left operand. If on one of the levels that must be created there is an object of a type different from Map then it is replaced by a new Map object.

In the native automata the variable access has O(1) complexity thanks to the fact that they are direct members of the automaton class. XML automata keep all the variables in an indexed list, the same way the states are stored, therefore the access algorithm complexity is O(n), where n is the length of the variable name hash [7, 8].

When comparing XML automata and native automata it can be inferred that the native ones must be used for intense processing and system calls and XML ones must be used for the business logic, control and code that must be transferred between applications. This way we can take advantage of both without sacrificing execution time or code modularity. The connection between the two types of automata is the messaging which ensures a uniform communication. Because Society framework was written using just the base platform for each

language it was implemented in, there are no additional dependencies for it to run. An advantage of this decision is the ease of extending the framework for the Android platform. In Android the Java code runs on a special virtual machine called Dalvik [9]. To run the intermediary Java code (Java byte-code) it is transformed into intermediary Dalvik code (Dalvik byte-code) for optimizations [10].

5 Conclusions

Communicative automata based programming makes the process of moving from the design to the implementation easier and the state or the activity diagrams to be found directly in the source code, without the need of a detailed documentation. The paradigm combines both imperative programming elements and declarative elements for obtaining a higher quality code with less effort. The new features it brings on top of the classic automata based programming, automaton atomicity and messaging communication, enforce practical rules with an aim for minimizing the errors: code modularization, data encapsulation at automaton level and request verification. These advantages have a great impact in production, especially in large projects where the work is assigned to a great number of programmers and architects.

Society framework has reached its purpose: the demonstration of the advantages brought by the communicative automata based programming. The differentiation between native automata and XML automata resulted in the development of two categories of automata, each with its advantages and disadvantages.

The Society framework, though it is not the most efficient implementation of the mechanisms in the communicative automata based programming, it draws closer to the industry needs. Large distributed applications or the ones that intensively use network transfers can be boosted by Society framework. Nonetheless, based on the application necessities, different mechanisms can be implemented for the automata. The target would be code optimization, security (message encryption, authentication, error tolerance, etc.) or the sending of messages through other environments (embedded systems, Bluetooth).

References

[1] E. Gribko. *Applications of Deterministic Finite Automata*. (2013).

[2] A. Shalyto. it Technology of Automata-Based Programming. (2004).

[3] J. Hopcroft, R. Motwani, J. Ullman. *Introduction to Automata Theory, Languages, and Computation, Second Edition*. Addison-Wesley, (2000).

[4] J. Edwards. *Multi-threading in JavaScript*. (2012).

[5] R. Gravelle. *Introducing HTML 5 Web Workers: Bringing Multi-threading to JavaScript*. (2012).

[6] S. Singhal. *Infix to Prefix Conversion. Sharing ideas, Sharing experiences*. (2012).

[7] Arno. *HashMap vs. TreeMap*. (2010).

[8] K. Normark. *Generic Dictionaries in C#*. (2010).

[9] J. Hildenbrand. *Android A to Z: What is Dalvik*. (2012).

[10] Security Engineering Research Group, Institute of Management SciencesPeshawar, Pakistan, *Analysis of Dalvik Virtual Machine and Class Path Library*. November (2009).

Andrei Micu, Adrian Iftene, Received July 20, 2015

Andrei Micu, Adrian Iftene
Institution: "Alexandru Ioan Cuza" University
Address: General Berthelot, No. 16
Phone: 004 - 0232 - 2011549
E–mail: andrei.micu@info.uaic.ro, adiftene@info.uaic.ro

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

On some trends in finite automata theory

Volodymyr V. Skobelev, Volodymyr G. Skobelev

Abstract

In this paper some results of research in two new trends of finite automata theory are presented. For understanding the value and the aim of these researches some short retrospective analysis of development of finite automata theory is given. The first trend deals with families of finite automata defined via recurrence relations on algebraic structures over finite rings. The problem of design of some algorithm that simulates with some accuracy any element of given family of automata is investigated. Some general scheme for design of families of hash functions defined by outputless automata is elaborated. Computational security of these families of hash functions is analyzed. Automata defined on varieties with some algebra are presented and their homomorphisms are characterized. Special case of these automata, namely automata on elliptic curves, are investigated in detail. The second trend deals with quantum automata. Languages accepted by some basic models of quantum automata under supposition that unitary operators associated with input alphabet commute each with the others are characterized.

Keywords: finite automata, finite rings, varieties, simulation, hash functions, elliptic curves, quantum automata.

1 Introduction

It is known that 'automaton' is one of the basic notions of computer science. Its significance was determined in the fundamental paper of A.M. Turing [1] where it was used as some formal model for informal notion of 'algorithm' (i.e. either a digital transducer, or an acceptor of a language). Foundations of finite automata (FA) theory were laid

in the middle of XX century [2]. In its essence any finite automaton is some formal model for processes that can be implemented on computers (due to restrictions on amount of memory).

Development of FA theory was motivated not only by its internal problems, but also it was carried out in close interaction with other areas of computer science. The last circumstance in many respects led to numerous applications of FA models. On the other hand, research of actual applied problems (including those in the area of information technologies) and emergence of some new paradigms for notion of 'computation' led to significant reconsideration problems in FA theory. As the result some entirely new sections of this theory began to appear.

In this paper we consider two of these sections. The first one deals with FA defined via recurrence relations over finite ring. The necessity of investigation of these models is substantially caused by the problems of modern cryptography [3, 4]. The second section deals with quantum FA, i.e. with some part of quantum algorithms theory which is developed intensively at present. Thus, quantum FA are based on the new paradigm of computations called 'quantum computations' [5, 6].

2 Survey of finite automata theory

The following two stages can be naturally highlighted in the development of FA automata theory.

The first stage covers 50s–80s of the XX century.

Finite automaton considered as a transducer was defined as a system $\mathtt{M} = (\mathtt{Q}, \mathtt{X}, \mathtt{Y}, \delta, \lambda)$ ($\mathtt{Q}$, $\mathtt{X}$ and $\mathtt{Y}$ are respectively finite set of states, finite input and finite output alphabets, $\delta : \mathtt{Q} \times \mathtt{X} \to \mathtt{Q}$ is the transition function and $\lambda : \mathtt{Q} \times \mathtt{X} \to \mathtt{Y}$ is the output function). Moore and Mealy models of FA and some their variants associated with FA functioning in time were determined. Problems of analysis and synthesis [7, 8, 9], the problem of completeness [10, 11] and problems of theory of experiments with FA [12] were investigated within these models. Analysis of transformations of free semigroups carried out by FA [13] had a significant influence on formation of algebraic theory of FA [14, 15] and automata-algebraic approach to software engineering [16]. It should

also be noted investigation of information-lossless FA [17, 18] which (possibly with some additional information) carry-out injective transformations of input semigroup into output semigroup. Just these FA demonstrate possibility for using of FA as some mathematical model for stream ciphers.

Finite automaton considered as an acceptor was defined as a system $\mathtt{M} = (\mathtt{Q}, \mathtt{X}, \delta, \mathtt{q}_{in}, \mathtt{Q}_{acc})$ ($\mathtt{Q}$ and $\mathtt{X}$ are respectively finite set of states and finite input alphabet, $\delta : \mathtt{Q} \times \mathtt{X} \to \mathtt{Q}$ is the transition function, $\mathtt{q}_{in} \in \mathtt{Q}$ is the initial state and $\mathtt{Q}_{acc} \subseteq \mathtt{Q}$ is the set of accepting states). An input string is accepted by $\mathtt{M}$ if it transforms the initial state into the set of accepting states. The set of such input strings is the language accepted by $\mathtt{M}$. It was proved that for any fixed finite alphabet the set of languages $\mathfrak{L}_{DFA}$ accepted by FA acceptors equals to the set of regular languages (Kleene's theorem). It should be noted that any FA acceptor is some 1-way 1-head Turing Machine (TM) with input tape, i.e. information is only read (1-way means that at every step the head of TM moves one cell to the right).

Non-deterministic FA acceptors were investigated under supposition that any subset $\mathtt{Q}_{in} \subseteq \mathtt{Q}$ of initial states could be chosen and any ternary relation $\delta \subseteq \mathtt{Q} \times (\{\Lambda\} \cup \mathtt{X}) \times \mathtt{Q}$ (Λ is the empty symbol) could define admissible transitions. Accepted language was defined as a set of strings that transform at least one initial state into the set of accepting states. It was proved that the set of all languages accepted by these acceptors equals to the set $\mathfrak{L}_{DFA}$. Although every non-deterministic FA acceptor can be effectively transformed into equivalent deterministic one, this transformation can lead to a significant increase in cardinality for the set of states (there are examples when non-deterministic FA acceptor has n states while equivalent deterministic one has 2^n states). Possibly, just this factor has grounded application of non-deterministic FA acceptors algebra [9] for formation of one of the main classes of discrete event systems designed to automate industrial process control.

Nontrivial generalization of non-deterministic FA acceptors was the emergence of probabilistic FA [19, 20]. In this model for each state and each input the probability of transition in each state was defined (thus, there is some deep inner link between probabilistic FA

and finite Markov chains [21]). Formally, probabilistic FA is a system $\mathtt{M} = (\mathtt{Q}, \mathtt{X}, \{M_{\mathtt{X}}\}_{\mathtt{x}\in\mathtt{X}}, \mathtt{u}_0, \mathtt{Q}_{acc})$, where $\mathtt{Q} = \{\mathtt{q}_1, \ldots, \mathtt{q}_n\}$ is the set of states, $\mathtt{X}$ is finite input alphabet, $M_{\mathtt{X}}$ ($\mathtt{x} \in \mathtt{X}$) is some stochastic $n \times n$-matrix of transitions, $\mathtt{u}_0 = (\alpha_1^{(0)}, \ldots, \alpha_n^{(0)})^T$ ($\alpha_i^{(0)} \in \mathbb{R}_+$ ($i \in \mathbb{N}_n$), $\sum\limits_{i=1}^{n} \alpha_i^{(0)} = 1$) is the initial distribution of states, and $\mathtt{Q}_{acc} \subseteq \mathtt{Q}$ is the set of accepting states. The evolution of $\mathtt{M}$ on input string $\mathtt{x}_1 \ldots \mathtt{x}_l$ ($l \in \mathbb{Z}_+$) is defined by identity $(\alpha_1^{(l)}, \ldots, \alpha_n^{(l)})^T = M_{\mathtt{x}_l} \ldots M_{\mathtt{x}_1} \mathtt{u}_0$. This string is accepted by $\mathtt{M}$ with probability $\mathtt{P}_{\mathtt{M}}(\mathtt{x}_1 \ldots \mathtt{x}_l) = \sum \alpha_i^{(l)}$, where the sum is over all i such that $\mathtt{q}_i \in \mathtt{Q}_{acc}$. Moreover, $\mathtt{M}$ accepts the language $\mathtt{L} \subseteq \mathtt{X}^+$ with: 1) probability p ($0.5 \leq p \leq 1$) if it accepts every string $\mathtt{w}_1 \in \mathtt{L}$ with probability not less than p, while any string $\mathtt{w}_2 \notin \mathtt{L}$ is accepted with probability not exceeding $1 - p$; 2) error $(p_1; p_2)$ ($0 \leq p_1 < p_2 \leq 1$) if it accepts every string $\mathtt{w}_1 \in \mathtt{L}$ with probability not less than p_2, while any string $\mathtt{w}_2 \notin \mathtt{L}$ is accepted with probability not exceeding p_1. It should be noted that any probabilistic FA is some 1-way 1-head probabilistic TM with input tape.

Progress in error-correcting codes development [22, 23] and linear systems analysis [24] stimulated research of FA presented via recurrence relations over finite fields [25].

The second stage in the development of FA theory started in 90s of the XX century.

Development of models for cryptographic protection of information had a great influence on FA theory. The following problems became actual. Firstly, it is analysis of preimages of output strings produced by FA [26]. Secondly, it is analysis of linear and polylinear recurrences over finite rings [27, 28]. These recurrences define some class of autonomous automata intended for design of generators of pseudo-random sequences used in modern ciphers. Thirdly, it is analysis of experiments with linear and bilinear automata defined via recurrence relations over finite fields [29]. Fourthly, it is investigation of complexity of FA identification [30]. This problem is caused by application of FA for analysis of computational security for stream ciphers [31, 32]. Fifthly, it is investigation of FA defined via algebraic recurrence relations over finite rings [33, 34]. If these FA are reversible, they can be

used as mathematical models for some stream ciphers.

The problems listed above show that formation of some new section of algebraic theory of FA is carried out at present. Essentially new factor for this section is the transition from transformations of free semigroups to transformations of algebraic structures performed by FA defined via recurrence relations over finite algebraic structures. These research are presented in chapter 3.

Since 1997 a variety of quantum FA (QFA) models different in capacity have been investigated. All of them are acceptors and they are defined in terms of 1-way k-head ($k \geq 1$) quantum TM (QTM) with input tape. Accepting of languages was analyzed both from the point 'with given probability' and 'with given error'.

Basic QFA models with measurement of a state only at the last step are listed below ($\mathtt{X}$ ($|\mathtt{X}| = m$) is input alphabet and with every letter $\mathtt{x} \in \mathtt{X}$ some unitary operator $U_{\mathtt{X}}$ acting in n-dimensional complex space $\mathbb{C}^n$ is associated).

The model MO-1QFA [35] is 1-way 1-head QTM $\mathtt{M} = (\mathtt{Q}, \mathtt{X}, |\varphi\rangle, \mathtt{Q}_{acc})$, where $\mathtt{Q} = \mathrm{B}_n$ ($\mathrm{B}_n = \{|i\rangle | i \in \mathbb{N}_n\}$) is the set of basic states, the unit vector $|\varphi\rangle \in \mathbb{C}^n$ is the pure initial state and $\mathtt{Q}_{acc} \subseteq \mathrm{B}_n$ is the set of accepting states. Probability that $\mathtt{M}$ accepts a string $\mathtt{w} = \mathtt{x}_1 \ldots \mathtt{x}_l \in \mathtt{X}^+$ equals to $\mathrm{P}(|\varphi\rangle, \mathtt{w}) = \| P_{acc} U_{\mathtt{w}} |\varphi\rangle \|^2$, where $U_{\mathtt{w}} = U_{\mathtt{x}_l} \ldots U_{\mathtt{x}_1}$ and P_{acc} is the projection operator on the subspace spanned by $\mathtt{Q}_{acc}$.

The model L-QFA [36] differs from the model MO-1QFA only that it deals with some initial mixed state $\{(|\varphi_i\rangle, \alpha_i)\}_{i \in \mathbb{N}_n}$ such that $|\varphi_i\rangle \in \mathbb{C}^n$ ($i \in \mathbb{N}_n$) are pair-wise different unit vectors, $\alpha_i > 0$ ($i \in \mathbb{N}_n$), and $\sum\limits_{i \in \mathbb{N}_n} \alpha_i = 1$ (α_i ($i \in \mathbb{N}_n$) is referred to as probability that at initial instant QTM $\mathtt{M}$ exists in the state $|\varphi_i\rangle$).

Probability that L-QFA $\mathtt{M}$ accepts a string $\mathtt{w} \in \mathtt{X}^+$ equals to $\mathrm{P}(\{(|\varphi_i\rangle, \alpha_i)\}_{i \in \mathbb{N}_n}, \mathtt{w}) = \sum\limits_{i \in \mathbb{N}_n} \alpha_i \mathrm{P}(|\varphi_i\rangle, \mathtt{w})$.

The model kQFA [37] is 1-way k-head QTM $\mathtt{M} = (\mathtt{Q}, \mathtt{T}, |\varphi\rangle, \mathtt{Q}_{acc})$ (at any instant all heads move simultaneously by one cell to the right), where $\mathtt{T} = \mathtt{X}^k \cup \bigcup\limits_{i=1}^{k-1} \mathtt{X}^i \{\Lambda\}^{k-i}$. It is worth to note that similarly to the case when the model L-QFA was defined as some generalization of

the model MO-1QFA, in [38] the model L-kQFA was defined as some generalization of the model kQFA.

Currently, analysis of QFA models is focused on detailed study of the set of accepted languages, as well as on resolving of the problem of identification of equivalent states. Some research of languages accepted by above listed models of QFA is presented in chapter 4.

3 Automata over algebraic structures

Let $\mathcal{K} = (K, +, \cdot)$ be fixed finite ring and $\mathcal{M} = \{\mathtt{M}_{\mathbf{a}}\}_{\mathbf{a}\in\mathbf{A}}$ ($\mathbf{A} \subseteq K^l$) be any family of FA

$$\mathtt{M}_{\mathbf{a}} : \begin{cases} \mathbf{q}_{t+1} = \mathbf{f}_1(\mathbf{q}_t, \mathbf{x}_{t+1}, \mathbf{a}) \\ \mathbf{y}_{t+1} = \mathbf{f}_2(\mathbf{q}_t, \mathbf{x}_{t+1}, \mathbf{a}) \end{cases} \quad (t \in \mathbb{Z}_+),$$

where $\mathbf{f}_1 : K^{n_1+n_2+l} \to K^{n_1}$ and $\mathbf{f}_2 : K^{n_1+n_2+l} \to K^{n_3}$ are fixed mappings, and $\mathbf{a}$ are parameters. It is known that via any experiment with an automaton $\mathtt{M}_{\mathbf{a}}$ the values of parameters $\mathbf{a} \in \mathbf{A}$ not always can be identified uniquely. So naturally arises the problem of design of some algorithm that simulates any $\mathtt{M}_{\mathbf{a}} \in \mathcal{M}$ with some accuracy (from the standpoint of cryptography this means 'an attack on the algorithm'). This problem has been resolved in [39, 40]. The essence of proposed solution is as follows.

We fix a set of parameters $\mathbf{B} \subseteq K^{l_1}$ and three families of mappings $\{\varphi^{(1)}_{\mathbf{b}} : K^{n_1+n_2} \to K^{n_3}\}_{\mathbf{b}\in\mathbf{B}}$, $\{\varphi^{(2)}_{\mathbf{b}} : K^{n_1} \times \bigcup_{j=1}^{r-1} (K^{n_3})^j \times K^{n_2} \to K^{n_3}\}_{\mathbf{b}\in\mathbf{B}}$ and $\{\varphi^{(3)}_{\mathbf{b}} : K^{n_1+rn_3+n_2} \to K^{n_3}\}_{\mathbf{b}\in\mathbf{B}}$. Let $\mathcal{G}_{\mathbf{B}} = \{G_{\mathbf{b}}\}_{\mathbf{b}\in\mathbf{B}}$ be the set of mappings, such that $G_{\mathbf{b}}(\mathbf{q}_0, \mathbf{x}_1 \dots \mathbf{x}_m) = \mathbf{y}_1 \dots \mathbf{y}_m$ ($\mathbf{b} \in \mathbf{B}, m \in \mathbb{N}$), where

$$\mathbf{y}_i = \begin{cases} \varphi^{(1)}_{\mathbf{b}}(\mathbf{q}_0, \mathbf{x}_1), & \text{if } i = 1 \\ \varphi^{(2)}_{\mathbf{b}}(\mathbf{q}_0, \mathbf{y}_1 \dots \mathbf{y}_{i-1}, \mathbf{x}_i), & \text{if } i = 2, \dots, r \\ \varphi^{(3)}_{\mathbf{b}}(\mathbf{q}_0, \mathbf{y}_{i-r} \dots \mathbf{y}_{i-1}, \mathbf{x}_i), & \text{if } r < i \le m \end{cases} .$$

Let $H_{\mathbf{b},\mathbf{q}_0}(\mathbf{x}_1 \dots \mathbf{x}_m) = G_{\mathbf{b}}(\mathbf{q}_0, \mathbf{x}_1 \dots \mathbf{x}_m)$ ($\mathbf{b} \in \mathbf{B}, \mathbf{q}_0 \in K^{n_1}, m \in \mathbb{N}$). It is evident that each family $\mathcal{H}_{\mathbf{b}} = \{H_{\mathbf{b},\mathbf{q}_0}\}_{\mathbf{q}_0\in K^{n_1}}$ ($\mathbf{b} \in \mathbf{B}$) defines

some finite automaton over the ring $\mathcal{K}$. Fixing surjection $h : \mathbf{A} \to \mathbf{B}$ we associate some family $\mathcal{H}_{h(\mathbf{a})}$ with every automaton $\mathtt{M}_{\mathbf{a}} \in \mathcal{M}$.

The ordered pair $(\mathcal{G}_{\mathbf{B}}, h)$ is defined as simulation model for the family $\mathcal{M}$. It is supposed that equalities $H_{h(\mathbf{a}),\mathbf{q}_0}\big|_{\bigcup_{i=1}^{r}(K^{n_2})^i} = F_{\mathbf{a},\mathbf{q}_0}\big|_{\bigcup_{i=1}^{r}(K^{n_2})^i}$ ($\mathbf{a} \in \mathbf{A}, \mathbf{q}_0 \in K^{n_1}$) hold, where $F_{\mathbf{a},\mathbf{q}_0} : (K^{n_2})^+ \to (K^{n_3})^+$ is the mapping realized by initial automaton $(\mathtt{M}_{\mathbf{a}}, \mathbf{q}_0)$. Semantics of these equalities is that simulation model $(\mathcal{G}_{\mathbf{B}}, h)$, connected to the input and the output channels of an automaton $\mathtt{M}_{\mathbf{a}}$ ($\mathbf{a} \in \mathbf{A}$) passes the first r output symbols, and then blocks the output channel of an automaton $\mathtt{M}_{\mathbf{a}}$ and simulates its behavior on the remaining tail of input string.

On the base of standard techniques of algorithms theory accuracy of simulation model $(\mathcal{G}_{\mathbf{B}}, h)$ has been defined for all combinations of notions 'in the worst case' and 'in average'. Asymptotically exact simulation models have been extracted and some sufficient conditions for existence of these models have been established in [39, 40].

It is evident that any hash function is some mapping of input semigroup into the set of states realized by some finite initial automaton. From the standpoint of cryptography analysis of hash functions families defined by outputless FA over finite ring is actual. This problem has been investigated in [41]. The main results are as follows.

Let $\mathcal{F}_{k,m}$ ($k \le m$) be the set of all mappings $\mathbf{f} : K^{k+m} \to K^k$, such that the following two equalities $|\{\mathbf{x} \in K^m | \mathbf{f}(\mathbf{q}, \mathbf{x}) = \mathbf{q}''\}| = |K|^{m-k}$ and $\{\mathbf{x} \in K^m | \mathbf{f}(\mathbf{q}, \mathbf{x}) = \mathbf{q}''\} \cap \{\mathbf{x} \in K^m | \mathbf{f}(\mathbf{q}', \mathbf{x}) = \mathbf{q}''\} = \emptyset$ hold for all $\mathbf{q}, \mathbf{q}', \mathbf{q}'' \in K^k$ ($\mathbf{q} \ne \mathbf{q}'$). It is evident that any mapping $\mathbf{f} \in \mathcal{F}_{k,m}$ defines strongly connected outputless automaton $\mathtt{M}_{\mathbf{f}}$, such that K^k is the set of states and K^m is input alphabet.

Let $H_{\mathbf{f},\mathbf{q}_0}$ be the mapping of input semigroup $(K^m)^+$ into the set of states K^k realized by initial automaton $(\mathtt{M}_{\mathbf{f}}, \mathbf{q}_0)$. Thus, automaton $\mathtt{M}_{\mathbf{f}}$ defines the family of hash functions $\{H_{\mathbf{f},\mathbf{q}_0}\}_{\mathbf{q}_0 \in K^k}$.

The following theorems are true:

Theorem 1. [41]. *For any mapping* $\mathbf{f} \in \mathcal{F}_{k,m}$ *if* $\mathbf{q}_0 \ne \mathbf{q}_0'$ $(\mathbf{q}_0, \mathbf{q}_0' \in K^k)$ *then* $H_{\mathbf{f},\mathbf{q}_0}(\mathbf{u}) \ne H_{\mathbf{f},\mathbf{q}_0'}(\mathbf{u})$ *for any input string* $\mathbf{u} \in (K^m)^+$.

Corollary 1. [41]. *For any* $\mathbf{f} \in \mathcal{F}_{k,m}$ *if* $\mathbf{q}_0 \ne \mathbf{q}_0'$ $(\mathbf{q}_0, \mathbf{q}_0' \in K^k)$ *then* $H^{-1}_{\mathbf{f},\mathbf{q}_0}(\mathbf{q}) \cap H^{-1}_{\mathbf{f},\mathbf{q}_0'}(\mathbf{q}) = \emptyset$ *for any* $\mathbf{q} \in K^k$.

Theorem 2. [41]. *For any mapping* $\mathbf{f} \in \mathcal{F}_{k,m}$ *and* $\mathbf{q}_0 \in K^k$ *equality* $|H^{-1}_{\mathbf{f},\mathbf{q}_0}(\mathbf{q}_t) \cap (K^m)^t| = |K|^{tm-k}$ $(\mathbf{q}_t \in K^k)$ *holds for all* $t \in \mathbb{N}$.

Let $p^{(1)}_{\mathbf{f},\mathbf{q}_0,t}(\mathbf{q})$ be probability that input string $\mathbf{u}$ randomly selected in the set $(K^m)^t$ is some solution of the equation $H(\mathbf{u}) = \mathbf{q}$, and $p^{(2)}_{\mathbf{f},\mathbf{q}_0,t}$ be probability that for two different input strings $\mathbf{u}$ and $\mathbf{u}'$ randomly selected in the set $(K^m)^t$ equality $H(\mathbf{u}) = H(\mathbf{u}')$ holds.

The following theorems are true:

Theorem 3. [41]. *For any mapping* $\mathbf{f} \in \mathcal{F}_{k,m}$ and $\mathbf{q}_0, \mathbf{q} \in K^k$ *equality* $p^{(1)}_{\mathbf{f},\mathbf{q}_0,t}(\mathbf{q}) = |K|^{-k}$ *holds for all* $t \in \mathbb{N}$.

Theorem 4. [41]. *For any mapping* $\mathbf{f} \in \mathcal{F}_{k,m}$ *and* $\mathbf{q}_0 \in K^k$ *equality* $p^{(2)}_{\mathbf{f},\mathbf{q}_0,t} = |K|^{-k}(1 - \frac{|K|^k - 1}{|K|^{mt} - 1})$ *holds for all* $t \in \mathbb{N}$.

Thus, the number $|K|^{-k}$ characterizes computing security for a family of hash functions $\{H_{\mathbf{f},\mathbf{q}_0}\}_{\mathbf{q}_0 \in K^k}$. This implies some feasibility for using these families in resolving problems of information protection.

Applications of elliptic curves over finite fields for resolving problems of information transformation justify feasibility of research FA defined on varieties (i.e. on the sets of solutions of systems of algebraic equations) over finite ring. It allows to set internal connections between modern algebraic geometry, systems theory, FA theory and cryptology.

From standpoint of algebraic FA theory and its applications it is reasonable to deal with the set $\mathcal{V}_1(\mathcal{K})$ of all varieties $\mathbf{V} \subseteq K^n$ with some algebra $(\mathbf{V}, \mathcal{F}_1 \cup \mathcal{F}_2)$, where $\mathcal{F}_1 = \{\alpha_0, \alpha_1, \ldots, \alpha_{k_1}\}$ and $\mathcal{F}_2 = \{\beta_1, \ldots, \beta_{k_2}\}$ are the sets of unary and binary operations, correspondingly. For any variety $\mathbf{V} \in \mathcal{V}_1(\mathcal{K})$ the algebra $(\mathbf{V}, \mathcal{F}_1 \cup \mathcal{F}_2)$ gives possibility to define the set $\mathcal{A}^{(1)}(\mathbf{V})$ of Mealy FA

$$\begin{cases} \mathbf{q}_{t+1} = \beta_{j_1}(\alpha_{i_1}(\mathbf{q}_t), \alpha_{x_{t+1}}(\mathbf{v}_1)) \\ \mathbf{y}_{t+1} = \beta_{j_2}(\alpha_{i_2}(\mathbf{q}_t), \alpha_{x_{t+1}}(\mathbf{v}_2)) \end{cases} \qquad (t \in \mathbb{Z}_+)$$

and the set $\mathcal{A}^{(2)}(\mathbf{V})$ of Moore FA

$$\begin{cases} \mathbf{q}_{t+1} = \beta_{j_1}(\alpha_{i_1}(\mathbf{q}_t), \alpha_{x_{t+1}}(\mathbf{v}_1)) \\ \mathbf{y}_{t+1} = \beta_{j_2}(\alpha_{i_2}(\mathbf{q}_{t+1}), \mathbf{v}_2) \end{cases} \qquad (t \in \mathbb{Z}_+),$$

where $\mathbf{v}_1, \mathbf{v}_2 \in \mathbf{V}$ are fixed points, $i_1, i_2 \in \mathbb{Z}_{k_1+1}$ and $j_1, j_2 \in \mathbb{N}_{k_2}$ are fixed integers, $\mathbf{q}_0 \in \mathbf{V}$, and $x_{t+1} \in \mathbb{Z}_{k_1+1}$ $(t \in \mathbb{Z}_+)$. Thus, for any $\mathtt{M} \in \mathcal{A}^{(1)}(\mathbf{V}) \cup \mathcal{A}^{(2)}(\mathbf{V})$ values of x_t, $\mathbf{q}_t$ and $\mathbf{y}_t$ are, correspondingly, an input symbol, a state and an output symbol at instant t.

Let $\mathbf{V}, \mathbf{U} \in \mathcal{V}_1(\mathcal{K})$. We say that: 1) the variety $\mathbf{U}$ is a homomorphic image of the variety $\mathbf{V}$, if the algebra $(\mathbf{U}, \mathcal{F}_1^{(2)} \cup \mathcal{F}_2^{(2)})$ is a homomorphic image of the algebra $(\mathbf{V}, \mathcal{F}_1^{(1)} \cup \mathcal{F}_2^{(1)})$; 2) varieties $\mathbf{U}$ and $\mathbf{V}$ are isomorphic if algebras $(\mathbf{U}, \mathcal{F}_1^{(2)} \cup \mathcal{F}_2^{(2)})$ and $(\mathbf{V}, \mathcal{F}_1^{(1)} \cup \mathcal{F}_2^{(1)})$ are isomorphic.

The next theorem is true:

Theorem 5. [42]. *Let $\mathbf{U}, \mathbf{V} \in \mathcal{V}_1(\mathcal{K})$. If $\mathbf{U}$ is a homomorphic image of $\mathbf{V}$, then there exist mappings $\Psi_j : \mathcal{A}^{(j)}(\mathbf{V}) \to \mathcal{A}^{(j)}(\mathbf{U})$ $(j = 1, 2)$, such that homomorphic image of any automaton $\mathtt{M}_j \in \mathcal{A}^{(j)}(\mathbf{V})$ is the automaton $\Psi_j(\mathtt{M}_j)$.*

Corollary 2. [42]. *Let $\mathbf{U}, \mathbf{V} \in \mathcal{V}_1(\mathcal{K})$. If $\mathbf{U}$ and $\mathbf{V}$ are isomorphic varieties, then there exist mappings $\Psi_j : \mathcal{A}^{(j)}(\mathbf{V}) \to \mathcal{A}^{(j)}(\mathbf{U})$ $(j = 1, 2)$, such that automata $\mathtt{M}_j \in \mathcal{A}^{(j)}(\mathbf{V})$ and $\Psi_j(\mathtt{M}_j)$ are isomorphic.*

Any elliptic curve γ over a finite field $\mathcal{K} = (K, +, \cdot)$ defines the abelian group $(G_\gamma, +_\gamma)$, where G_γ is the set of all points of γ including specified point $\mathcal{O}$ (this point serves as the neutral element of the group). Setting $\mathcal{F}_1 = \{\alpha_0, \alpha_1, \ldots, \alpha_{k_1}\}$ $(1 \le k_1 < |G_\gamma|)$, where $\alpha_0(P) = \mathcal{O}$ $(P \in G_\gamma)$ and $\alpha_i(P) = \underbrace{P +_\gamma \ldots +_\gamma P}_{i \text{ times}}$ $(P \in G_\gamma)$ for all $i = 1, \ldots, k_1$, and $\mathcal{F}_2 = \{+_\gamma\}$, we get some algebra $(G_\gamma, \mathcal{F}_1 \cup \mathcal{F}_2)$. Thus, any elliptic curve γ defines some variety of above considered type.

For any $P_1, P_2 \in G_\gamma \backslash \{\mathcal{O}\}$ and $n, m \in \mathbb{N}_{k_1}$ recurrence relations

$$\begin{cases} q_{t+1} = nq_t +_\gamma x_t P_1 \\ y_{t+1} = mq_t +_\gamma x_t P_2 \end{cases} \quad (t \in \mathbb{Z}_+)$$

and

$$\begin{cases} q_{t+1} = nq_t +_\gamma x_t P_1 \\ y_{t+1} = mq_{t+1} \end{cases} \quad (t \in \mathbb{Z}_+),$$

where $x_{t+1} \in \mathbb{N}_{k_1}$, define the family $\mathcal{M}_{1,\gamma,k_1}$ of Mealy FA and the family $\mathcal{M}_{2,\gamma,k_1}$ of Moore FA, correspondingly.

The following theorems are true:

Theorem 6. [43]. *For any automaton* $\mathtt{M}_1 \in \mathcal{M}_{1,\gamma,k_1}$ *identification of its initial state (with the accuracy to the set of equivalent states) is reduced to searching any solution of equation* $mu = a_0$, *where an element* $a_0 \in G_\gamma$ *is determined as the result of some simple experiment of the length* 1 *with the automaton* $\mathtt{M}_1$.

Theorem 7. [43]. *For any automaton* $\mathtt{M}_2 \in \mathcal{M}_{2,\gamma,k_1}$ *identification of its initial state (with the accuracy to the set of equivalent states) is reduced to searching any solution of equation* $mnv = b_0$, *where an element* $b_0 \in G_\gamma$ *is determined as the result of some simple experiment of the length* 1 *with the automaton* $\mathtt{M}_2$.

Theorem 8. [43]. *Exact imitation model for the family* $\mathcal{M}_{1,\gamma,k_1}$ *of Mealy FA can be designed as the result of some multiple experiment of the multiplicity* 3 *and of the height not exceeding* $|G_\gamma| + 1$. *The total length of all input strings applied to the investigated automaton in this experiment does not exceed* $|G_\gamma| + 1 + 0.5|G_\gamma| \cdot (|G_\gamma| + 3)$.

Theorem 9. [43]. *Exact imitation model for the family* $\mathcal{M}_{2,\gamma,k_1}$ *of Moore FA can be designed as the result of some multiple experiment of the multiplicity* 2 *and of the height not exceeding* $|G_\gamma|$. *The total length of all input strings applied to the investigated automaton in this experiment does not exceed* $|G_\gamma| + 0.5|G_\gamma| \cdot (|G_\gamma| + 1)$.

These results imply some feasibility for using above considered families of FA in resolving problems of information protection.

4 Quantum Automata

QFA under supposition that unitary operators associated with input alphabet commute each with the others have been investigated in [44]. Languages accepted either with given probability, or with given error have been characterized as follows.

Let $\mathtt{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_m\}$ be the input alphabet of QFA. It is supposed that elements of the set $\mathcal{U} = \{U_i | i \in \mathbb{N}_m\}$ (U_i is unitary operator associated with $\mathtt{x}_i \in \mathtt{X}$) commute each with the others. With any input string $\mathtt{w} \in \mathtt{X}^l$ ($l \in \mathbb{N}$) the string $pr_{\mathcal{U}}(\mathtt{w}) = U_1^{r_1} \ldots U_m^{r_m}$ can be associated,

where r_i ($i \in \mathbb{N}_m$) is the number of occurrences of $\mathtt{x}_i$ in $\mathtt{w}$. Let $\equiv_{\mathtt{X},\mathcal{U}}$ be equivalence on $\mathtt{X}^+$ defined as follows: $\mathtt{w}_1 \equiv_{\mathtt{X},\mathcal{U}} \mathtt{w}_2 \Leftrightarrow pr_{\mathcal{U}}(\mathtt{w}_1) = pr_{\mathcal{U}}(\mathtt{w}_2)$.

The following theorem is true:

Theorem 10. [44]. *Let $\mathcal{U}$ be any set of unitary operators that commute each with the others. Then any language accepted (either with given probability, or with given error) by the model MO-1QFA, as well as by the model L-QFA with measurement at final instant only is union of some elements of the factor-set $\mathtt{X}/\equiv_{\mathtt{X},\mathcal{U}}$.*

Similar results can be established for models kQFA and L-kQFA under supposition that unitary operators associated with elements of the set $\mathtt{X}^k$ commute each with the others. However, some technical difficulties arise with definition of equivalence on the set $\mathtt{T}$ due to the presence of elements of the set $\bigcup_{i=1}^{k-1} \mathtt{X}^i\{\Lambda\}^{k-i}$.

In [38] presented above approach has been worked out in detail for one of the most simple models of QFA, namely 1-qubit QA under supposition that associated unitary operators are rotations of the Bloch sphere [5, 6] around the y-axe and measurement of a state is produced at final instant only. Criteria when investigated models MO-1QFA, L-QFA, kQFA and L-kQFA accept some language with given probability, as well as with given error has been established.

These results imply feasibility of investigation of the structure of the set $\mathfrak{S}$ of all finitely generated commutative semigroups of special unitary operators in $\mathbb{C}^2$ (the notion 'special' means that the determinant of a matrix that defines unitary operator equals to unit). This problem has been investigated in [45]. Six sets of semigroups contained in the set $\mathfrak{S}$ have been extracted. However, it is still unknown, if these sets cover the set $\mathfrak{S}$ or not.

5 Conclusions

In given paper some research in two new trends of FA theory have been presented.

The first trend deals with investigation of FA families defined on algebraic structures over finite rings. Presented results justify some

feasibility for using these families in resolving problems of information protection. Based on this viewpoint, the following further research can be pointed. Firstly, searching non-trivial FA families for which any asymptotically accurate simulation model is much more complicated than a system of equations defining the family itself. Secondly, characterization of families of reversible FA for which transition to any simulation model results in essential loss of accuracy. Thirdly, detailed investigation into computational security of specific families of hash-functions determined by outputless automata over finite rings. Fourthly, detailed investigation into computational security of FA families defined on elliptic curves over finite fields.

The second trend deals with investigation of languages accepted by QFA models under supposition that unitary operators associated with input alphabet commute each with the others. In this direction, some progress in investigation of 1-qubit QFA have been achieved. However, no similar results are known for l-qubit QFA ($l \geq 2$). Possibly, the reason is that no visual geometric model which is similar to Bloch sphere is known for $l \geq 2$. Characterization of l-qubit QFA ($l \geq 2$) under supposition that unitary operators associated with input alphabet commute each with the others forms some trend for future research.

References

[1] A.M. Turing. *On computable numbers, with an application to the Entscheidungsproblem.* Proc. London Math. Soc., ser. 2, vol. 42 (1936), pp. 230–265.

[2] *Automata studies* (Ed. by C.E. Shannon, J. McCarthy). Princeton University Press, 1956.

[3] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone. *Handbook of applied cryptography.* CRC Press, 2001.

[4] J. Kaz, Y. Lindell. *Introduction to modern cryptography.* CRC Press, 2007.

[5] M.A. Nielsen, I.L. Chuang. *Quantum computation and quantum information.* Cambrige University Press, 2010.

[6] C.P. Williams. *Explorations in quantum computing.* Springer-Verlag London Limited, 2011.

[7] V.M. Glushkov. *Synthesis of digital automata.* Moskow, Nauka, 1962. [in Russian]

[8] Z. Kohavi. *Switching and finite automata theory.* New York, McGraw-Hill, 1970.

[9] B.A. Trachtenbrot, Y.M. Barzdin. *Finite automata. Behavior and synthesis.* North-Holland, 1973.

[10] A.A. Letichevskii. *Completeness conditions for finite automata.* USSR Computational Mathematics and Mathematical Physics, vol. 1, issue 3 (1962), pp. 829–840.

[11] M.I. Kratko. *Undecidability of completeness for finite automata.* Doklady AN SSSR, vol. 155, No 1 (1964), pp. 35–37. [in Russian]

[12] A. Gill. *Introduction to the theory of finite-state machines.* New York, McGraw-Hill, 1962.

[13] V.M. Glushkov. *The abstract theory of automata.* Russian Mathematical Surveys, vol. 16, No 5 (1961), pp.1–53.

[14] S. Eilenberg. *Automata, languages and machines. Vol. A.* New York, Academic Press, 1974.

[15] S. Eilenberg. *Automata, languages and machines. Vol. B.* New York, Academic Press, 1976.

[16] V.M. Glushkov, G.E. Tseitlin, E.L. Yushchenko. *Algebra, languages, programming.* Kiev, Naukova Dumka, 1978. [in Russian]

[17] D.A. Huffman. *Canonical forms for information-lossless finite state logical machines.* IRE Transactions Circuit Theory. Special Supplement, vol. CT-6 (1959), pp. 41–59.

[18] S. Even. *On information-lossless automata of finite order.* IEEE Transactions on Electronic Computers, vol. EC: 14, Issue: 4 (1965), pp. 561–569.

[19] M.O. Rabin *Probabilistic automata.* Information and Control, No 3 (1963), pp. 230–245.

[20] A. Paz A. *Introduction to probabilistic automata.* New York, Academic Press, 1971.

[21] J.G. Kemeny, T.L. Snell. *Finite Markov chains.* Princeton, NJ: D. Van Nostrand, 1960.

[22] E.R. Berlekamp. *Algebraic coding theory.* New York, McGraw-Hill, 1968.

[23] W.W. Peterson, E.J. Weldon, Jr. *Error-correcting codes.* The M.I.T. Press, Cambridge, MA, 1972.

[24] L.A. Zadeh, C.A. Desoer. *Linear system theory.* New York, McGraw-Hill, 1963.

[25] A. Gill. *Linear sequential circuits - analysis, synthesis, and applications.* New York, McGraw-Hill, 1966.

[26] B.A. Sevastyanov, V.P. Chistyakov. *On the number of input sequences corresponding to the output sequences of a finite automaton.* Review of Applied and Industrial Mathematics, vol. 1, Moskow, TVP (1994), pp. 96–107. [in Russian]

[27] V.L. Kurakin, A.S. Kuz'min, A.A. Nechaev. *Pseudo-random and polylinear sequences.* Memoires in Discrete Mathematics, vol. 1, Moskow, TVP (1997), pp. 139–202. [in Russian]

[28] V.L. Kurakin, A.S. Kuz'min, A.A. Nechaev. *Properties of linear and polylinear recurrencies over Galois rings (I).* Memoires in Discrete Mathematics, vol. 2, Moskow, TVP (1998), pp. 191–222. [in Russian]

[29] D.V. Speransky. *Experiments with linear and bilinear finite automata.* Saratov, Saratov State University, 2004. [in Russian]

[30] A.V. Babash. *Approximate models for finite automata.* Review of Applied and Industrial Mathematics, vol. 12 (2005), pp. 108–117. [in Russian]

[31] N. Courtois, W. Meier. *Algebraic attack on stream ciphers with linear feedback.* LNCS, vol. 2656 (2003), pp. 345–349.

[32] V.N. Trenkaev, R.G. Kolesnikov. *Automata approach to attack on symmetric ciphers.* Bulletin of Tomsk State University. Appendix, No 23, (2007), pp. 130–135.

[33] V.V. Skobelev, V.G. Skobelev. *Ciphersystems analysis.* Donetsk, IAMM of NASU, 2009. [in Russian].

[34] V.V. Skobelev, N.M. Glazunov, V.G. Skobelev. *Varieties over rings. Theory and applications.* Donetsk, IAMM of NASU, 2011. [in Russian].

[35] C. Moore, J. Crutchfield. *Quantum automata and quantum grammars.* Theor. Comput. Sci., vol. 237 (2000), pp. 257–306.

[36] A. Ambainis, M. Beaudry, M. Golovkins, at al. *Algebraic results on quantum automata.* LNCS, vol. 2996 (2004), pp. 93–104.

[37] A. Belovs, A. Rosmanis A., J. Smotrovs. *Multi-letter reversible and quantum finite automata.* LNCS, vol. 4588 (2007), pp. 60–71.

[38] V.G. Skobelev. *Analysis of finite 1-qubit quantum automata unitary operators of which are rotations.* Visn., Ser. Fiz.-Mat. Nauky, Kyiv. Univ. Im. Tarasa Shevchenka, No. 2 (2014), pp. 194–201.

[39] V.V. Skobelev. *Simulation of automata over a finite ring by the automata with a finite memory.* Journal of Automation and Information Sciences, vol. 44, issue 5 (2012), pp. 57–66.

[40] V.V. Skobelev. *Analysis of the problem of recognition of automaton over some ring.* Dopov. Nats. Akad. Nauk Ukr., Mat., Pryr., Tekh. Nauky, No 9 (2012), pp. 29–35.

[41] V.V. Skobelev. *Analysis of families of hash functions defined by automata over a finite ring.* Cybern. Syst. Anal., vol. 49, No. 2 (2013), pp. 209–216.

[42] V.V. Skobelev. *Analysis of automata models determined on varieties ovef finite ring.* Journal of Automation and Information Sciences, vol. 45, issue 8 (2013), pp. 21–31.

[43] V.V. Skobelev. *Automata on algebraic structures.* Donetsk, IAMM of NASU, 2013. [in Russian]

[44] V.G. Skobelev. *Quantum automata with operators that commutes.* Visn., Ser. Fiz.-Mat. Nauky, Kyiv. Univ. Im. Tarasa Shevchenka, Special Issue (2013), pp. 34–41.

[45] V.G. Skobelev. *On the structure of the set of all finitely generated semigroups of special unitary operators in the space* $\mathbb{C}^2$. Visn., Ser. Fiz.-Mat. Nauky, Kyiv. Univ. Im. Tarasa Shevchenka, No. 3 (2014), pp. 182–187.

Volodymyr V. Skobelev, Volodymyr G. Skobelev Received June 20, 2015

Volodymyr V. Skobelev
V.M. Glushkov Institute of Cybernetics of NAS of Ukraine
40 Glushkova ave., Kyiv, Ukraine, 03187
Phone: +38 063 431 86 05
E–mail: vvskobelev@incyb.kiev.ua

Volodymyr G. Skobelev
V.M. Glushkov Institute of Cybernetics of NAS of Ukraine
40 Glushkova ave., Kyiv, Ukraine, 03187
Phone: +38 063 431 86 05
E–mail: skobelevvg@mail.ru

Part 7

Semantic technologies

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

The Law of Gravitation in Ontology Dynamics

(Invited paper)

Vadim Ermolayev

Abstract

This paper presents the substance of the invited talk given at the 2015 Workshop on Foundations of Informatics. The idea undelying the presented research work is to borrow a plausible analogy of a "system law" from the field of Dynamics in Mechanics – the Newton's Law of Universal Gravitation. This analogy is exploited for building the law of gravitation in dynamic systems comprising a Domain of Discourse and knowledge representations (ontologies) describing this domain. As the elements of knowledge representation do not possess physical mass, this component of the gravitation law is substituted by the fitness of an ontology to the requirements of the knowledge stakeholders relevant for the described domain. It is also argued in the paper that the implementation of the developed theoretical framework is feasible as the supporting techniques, including some software tools, already exist. As the examples of the relevant component methods and tools, the paper presents the OntoElect methodology, Ontology Difference Visualizer, and Structural Difference Discovery Engine. These instruments help solve the practical problems in eliciting domain requirements, developing structural contexts for the requirements, generating the mappings between these structural contexts and the target ontology, computing increments and decrements of ontology fitness based on these mappings. It is concluded that the presented framework has prospects to be applied practically for visualization and analysis of ontology changes in dynamics. Use cases in ontology refinement and anomaly detection are suggested for validation.

Keywords: Ontology, Domain, Dynamics, Gravitation, Fitness

1 Introduction

The world of knowledge representations, comprising ontologies, is by its nature a reflection of the world we live in. Dynamics in physical, social, biological contexts are the subject of study by several disciplines, where useful analogies can be sought. The findings hint about a way to identify and specify useful aspects and help offer the law to describe dynamics in ontological systems.

It is known for example from Mechanics, the branch of Physics and Engineering, that Kinematics studies the motion of objects without direct reference to the causes of this motion. Motion in this context is understood as a change of position, often compared to a reference point. In difference to Kinematics, Dynamics is concerned with forces and torques and their effect on the motion of objects. For example, in Dynamics it is analyzed why an object changes its position and due to which causes or influences the acceleration has this specific value function over time.

One of the particular kinds of forces of interest regarding a physical system is gravitation. Basically, gravitation forces are known to be expressed by the Newton's Law of Universal Gravitation [1] as proportional to the product of interacting masses and inverse to the square distance between these masses. In biological and social systems similar "forces" reflect the degree of "attraction" of a particular object to a group, habitat, etc. For knowledge representations, an analogy to the notions of mass, gravitation, force could be sought in terms of the fitness of a knowledge representation module to the requirements of the stakeholders in the Domain of Discourse or its similarity to the other modules which could be found regarding the Domain of Discourse.

This paper presents the substance of the invited talk given at the 2015 Workshop on Foundations of Informatics (FOI-2015). It starts with the discussion of the notion of an ontology – one of the fundamental concepts in Knowledge Representation and Management. In this context, the property of being a "shared conceptualization" is explained in terms of the fitness to the requirements of the domain knowledge stakeholders, resulting in their commitment. The paper

continues with an outline of the state of the play in the field of Ontology Change, putting a particular emphasis on ontology Dynamics versus Kinematics. Then, the fundamentals of the theory of Ontology Dynamics based on the analogy to the Newton's Law of Universal Gravitation are presented. Yet further, the paper deliberates about the techniques for implementing this theoretical ontology gravitation framework. The paper concludes with the summary of the presented work and outlines the potential applications of the presented framework in Ontology Refinement and Anomaly Detection.

2 Ontologies, Domain Requirements, Fitness, and Dynamics

An ontology is often denoted as a "formal, explicit specification of a shared conceptualization" (c.f. [2]) and this paper follows this definition. In particular it is focused on describing and exploiting the properties of being "formal" and "explicit" regarding the representation of a conceptualization (specification), and – even more importantly – the property of being "shared" regarding the conceptualization itself. It is also emphasized that the completeness of an ontology has a straightforward impact on becoming a "shared conceptualization".

Being "formal" means that an ontology has to be specified using a formally defined ontology specification language such that logical inference is enabled with respect to this artifact. To enable logical inference, such a language needs to be based on logics – so an ontology is a logical theory. Ontology is also a descriptive theory as it is developed with the purpose to describe common sense, abstract high-level notions, or a Domain of Discourse.

Following for example [3], an ontology is a logical descriptive theory formally denoted as a tuple $O = \langle C, P, I, T, V, \leq, \perp, \in, = \rangle$ where C is the set of *concepts* (or *classes*); P is the set of *properties* (*object* and *datatype properties*); I is the set of *individuals* (or *instances*); T is the set of *datatypes*; V is the set of *values*; $\leq$ is a *reflexive*, *anti-symmetric*, and *transitive relation* on $(C \times C) \cup (P \times P) \cup (T \times T)$ called *specializa-*

tion, that helps form partial orders on C and P called *concept hierarchy* and *property hierarchy* respectively; $\perp$ is an *irreflexive* and *symmetric relation* on $(C \times C) \cup (P \times P) \cup (T \times T)$ called *exclusion*; $\in$ is a relation over $(I \times C) \cup (V \times P)$ called *instantiation*; $=$ is a relation over $I \times P \times (I \cup V)$ called *assignment*. The sets C, P, I, T, V are pairwise disjoint. It is also assumed (c.f. [4]), that an ontology O comprises its schema S and the assertional part A:

$$O = \langle S, A \rangle ; S = \langle C, P, T \rangle ; A = \langle I, V \rangle . \tag{1}$$

Ontology schema S is also referred to as a terminological component (TBox). It contains the statements describing the concepts of O, the properties of those concepts, and the axioms over the schema constituents. The set of individuals A, also referred to as assertional component (ABox), is the set of the ground statements about the individuals and their attribution to the schema – i.e. where these individuals belong.

This paper focuses on the ontologies that describe a particular well circumscribed Domain of Discourse – classified as domain ontologies. The reason for this emphasis is that any ontology development process, including its change management or refinement, takes as an input the requirements by the subject experts in the domain of interest and produces the ontology as its output – covering those requirements correctly and to the maximal possible extent. Straightforwardly, the set of methods shaping out this process needs to comprise the mechanisms for:

- Eliciting the (change[1]) requirements from the domain knowledge stakeholders as fully as possible
- Measuring how completely the requirements were captured
- Transforming the elicited requirements to the (changes in the) ontology

[1] Change requirements are elicited in the ontology Refinement phase. In the phase of Initial Development initial requirements are collected.

- Measuring how well the result fits to the intentions of the domain knowledge stakeholders

If a methodology fails to do any of the above sufficiently well, then the commitment of the knowledge stakeholders to the output ontology will be low. So, such a product cannot be regarded as a really "shared conceptualization".

Hence, a domain ontology O_D could be regarded as a harmonized formal, and explicit representation of the union of the interpretations (K) by the knowledge stakeholders $s_i \in S$ of the subject domain D. So, naïvely, we may elicit all the K-s and build the ontology of those as:

$$O_D = hrm(\bigcup_S unf_{f_j}(K_{s_i})), \tag{2}$$

where hrm is a harmonization function and unf is the transformation that maps a knowledge interpretation represented in the form f_j to the knowledge representation formalism used by the knowledge engineer (unification). Even if so, harmonization and unification functions are not easy to perform. For example, a formalism f_j for K_{s_i} could be more expressive than the ontology specification language used for coding O_D; K_{s_m} and K_{s_n} could be mutually contradictory in some parts; etc. Reality introduces more complications – mainly influencing the properties of being explicit and complete:

- K-s are **subjective**. The stakeholders interpret their domain based on their individual background knowledge and experience.
- K-s are **tacit**. The views on the domain by the subject experts are often not stated explicitly. On the contrary, some parts of those K-s are assumed, taken as evident or default, subsuming that (all) the professional community regards these assumptions in a similar way. The tacit parts are the cause for difference in interpretations, or even misinterpretations.
- K-s are **partial**. Subject experts focus on their narrow context of professional interest and expertise, and have only a shallow

coverage of the broader area within the domain. The partiality and fragmentation of their K-s is the reason for (a) contradictions between different views on the overlapping contexts; and (b) gaps in the coverage of the domain.

- K-s are **not available**. The knowledge stakeholders are not readily willing to spend their time for materializing their K-s or revealing them to knowledge engineers in another form.

In Ontology Engineering and Management the degree of the conformance of an ontology to the requirements of the domain knowledge stakeholders is regarded as its *fitness*. Measuring ontology fitness is not an easy task as one has to have: the requirements; the ontology; these two compared and difference measured. Several approaches to ontology fitness measurement are known from the literature – e.g. [5, 6]. One of these approaches has been developed as a part of the OntoElect ontology engineering methodology [7]. In OntoElect, ontology fitness to domain knowledge stakeholder requirements is understood as proportional to the ratio of positive and negative votes of these stakeholders regarding the assessed ontology. These votes are collected indirectly [7], as for example in [8], by:

- Extracting a saturated set of multi-word key terms from the statistically representative document corpus
- Detecting the most influential key terms by applying weights to the most "important" documents in the corpus
- Transforming the natural language definitions of the selected key terms to formalized structural contexts in the ontology specification language; and
- Mapping the structural contexts to the ontology

Ontologies describing realistic domains could be substantially large and complex in their structures and properties. So, the development and management of these descriptive theories call for solving several interesting research problems. As profoundly surveyed in [9], ontology change – changing an ontology in response to a certain need – is one of

the most important and challenging among them. The term of ontology change is often used broadly – to cover several interrelated facets of the problem and comprise different kinds of changes to ontologies: in response to external events; caused by translations to a different language having different expressive power; caused by the evolution of stakeholder requirements; introduced by the ontology engineer according to the evolved understanding of the domain; etc. Several research sub-fields have emerged to cope with this broad variety of change aspects: ontology evolution; versioning; merging; mapping; matching; alignment; refinement – to mention the most prominent.

The plethora of these research facets, all looking at the phenomenon of change in ontologies, gave also the birth to the Ontology Dynamics community (http://ontologydynamics.org/). It may be noticed however, that the mainstream approach, also adopted by the aforementioned community, follows more Kinematics than Dynamics. Indeed, the term of "ontology change" is referred to "the problem of deciding the modifications to perform upon an ontology in response to a certain need for change as well as the implementation of these modifications and the management of their effects in depending data, services, applications, agents or other elements" (c.f.[9]). In simple words: given the need for a change, it is decided what is changed and to what extent – i.e. if following the analogy with Mechanics, how much the position, velocity, acceleration of the object changes.

It appears that the field of Ontology Change does not look sufficiently deeply into the causes of a change – which is in fact the task for Ontology Dynamics. In this paper some steps are made toward laying out a foundation for filling this gap based on analyzing the (changes in the) fitness of an ontology to a particular Domain of Discource.

3 Ontology Dynamics and the Law of Gravitation

Let us now think of a system, comprising a Domain of Discourse and several ontologies describing it, as of a closed "mechanical" system. For making this analogy plausible – i.e. to be able to propose usable

dynamic laws – we have to find the proper analogies to the mechanical notions of: a coordinate grid and its origin; a position, a distance, a motion; a mass; and a force (gravitation).

Let us assume that a Domain of Discourse (D) is adequately modeled by the set of all relevant requirements (R), by its knowledge stakeholders, for representing knowledge in this domain. For building a grid based on these requirements it is assumed, as pictured in Fig. 1a, that:

- All the requirements are placed in the centre of the D; and
- They are not equal in their importance – i.e. have different spheres of influence around the centre of gravitation, which is quantified using normalized scores $ns \in [0, 1]$.

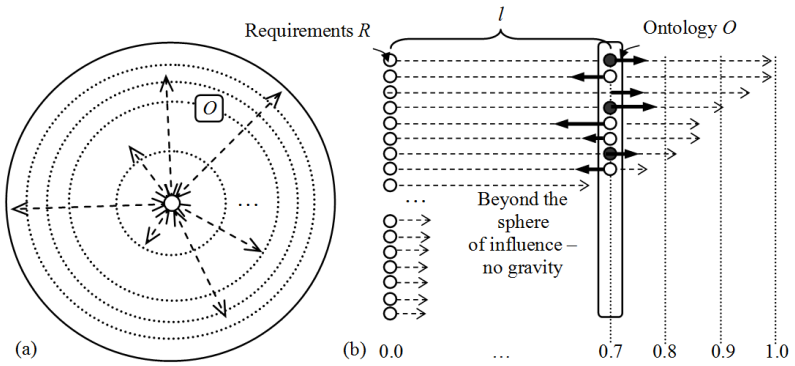


Figure 1. Domain requirements, their spheres of influence (a), and gravitation forces (b)

Imagine now that an ontology (O) appears to be positioned in D at a (semantic) distance l from its centre (Fig. 1(b)). This can be any location on the circle of radius l around the centre of the grid. We are now interested in what might be the forces influencing O in this position.

Let us assume that O is checked against the requirements r from R which spheres of influence reach the position of O (i.e. $ns_r \geq l$). The following are the possible outcomes of these checks:

- A particular part of O, say a semantic context $o \in O$ (a white coloured circle in Fig. 1(b)), fulfils the requirement r. Therefore O becomes more fitting to R. In this case we will consider that the increase in fitness ($\Delta\Phi_o^+$) creates a positive gravitation force $\overrightarrow{G_o^+}$ applied to O and directed towards the centre of D, as pictured in Fig. 1(b). The absolute value of this force is computed using a direct analogy with the Newton's Law of Universal Gravitation [1]:

$$G_o^+ = \frac{1 \times \Delta\Phi_o^+}{(ns_r)^2}, \tag{3}$$

where: "1" in the numerator is the fitness of r with respect to D – meaning that r fits D perfectly as one of its requirements; the value of $\Delta\Phi_o^+$ is within $[0, 1]$.

- There is no semantic context $o \in O$ that fulfills the requirement r (no circle on the ontology side in Fig. 1(b)) or there is an o that contradicts r (a black coloured circle in Fig. 1(b)). In both cases O becomes less fitting to R. Therefore we will consider that the decrease in fitness ($\Delta\Phi_O^-$ for a missing semantic context; $\Delta\Phi_o^-$ for a context contradictory to o) creates a negative gravitation force $\overrightarrow{G_O^-}$ or $\overrightarrow{G_o^-}$ applied to O and directed towards the periphery of D, as pictured in Fig. 1(b). Similarly to (3), the absolute values of these forces are computed as:

$$\begin{aligned} G_O^- &= \frac{1 \times \Delta\Phi_O^-}{(ns_r)^2}, \\ G_o^- &= \frac{1 \times \Delta\Phi_o^-}{(ns_r)^2}. \end{aligned} \tag{4}$$

The overall gravitation force applied to O as an influence by D is computed as a vector sum:

$$\overrightarrow{G_O}\Big|_D = \sum_{r \in R : ns_r \geq l} \left(\overrightarrow{G_o^+} + \overrightarrow{G_O^-} + \overrightarrow{G_o^-} \right). \tag{5}$$

O is considered as *properly positioned* within D when it reaches its *equilibrium* state with respect to the gravitation field in D, i.e. appears at a distance l from the centre of D at which $\overrightarrow{G_O}\Big|_D = \overrightarrow{0}$. This distance could be interpreted as an integral measure of the semantic difference between what does O describe and what is required to be described for D by its knowledge stakeholders. If O is not in an equilibrium state regarding D, $\overrightarrow{G_O}\Big|_D$ will cause it to move either towards the centre of D or towards its periphery.

O also generates its gravitation field which affects D. However, we do not take into account the movement of D because the centre of the grid (and therefore a potential observer) is always located in the centre of D. The gravitation field of O will come into effect in this grid if there are several ontologies positioned within D. This case is resolved similarly to the case of a single ontology described above. Ontology A reaches its *equilibrium state* within D and with respect to the ontologies B and C if $\overrightarrow{G_A}\Big|_D + \overrightarrow{G_A}\Big|_B + \overrightarrow{G_A}\Big|_C = \overrightarrow{0}$. So do the other ontologies B and C. In this equilibrium state the distances l_{AB}, l_{AC}, l_{BC} could be interpreted as the integral measures of the semantic difference in the respective pairs of ontologies, also under the influence of R in D. One topical difference for the case of multiple ontologies is that the differences and similarities in the pairs of ontologies are computed differently compared to the fitness in the pair O, D. For comparing ontologies, the use of matching techniques is the mainstream approach.

The subsequent section elaborates how could the set of requirements R be formed for D and also how could fitness changes and semantic differences between ontologies be computed.

4 Supporting Techniques

For making the theoretical framework based on the Law of Gravitation usable in practice several technical problems have to be solved and corresponding software tools be developed. The techniques and tools applicable in this context are presented in this section.

4.1 Extracting Domain Requirements

As explained in Section 2, a feasible way to make domain requirements explicit is to do it indirectly – by extracting multi-word key terms from a representative corpus of documents describing the domain. A document corpus could be considered as representative if it is sufficiently completely covers the description of the domain. One way to assess completeness is to use the *saturation* metric proposed in OntoElect.

Let $Doc = Doc_1, ..., Doc_{i+1} = Doc_i \bigcup \Delta_{i+1}, ..., Doc_n$ be the sequence of the samples of the document corpus which are built incrementally – i.e. each subsequent sample Doc_{i+1} in the sequence is created by adding a number of new relevant documents (Δ_{i+1}) to the previous sample Doc_i. Let $T_i = \{(t^i_j, s^i_j, ns^i_j)\}$ be the bag of terms and their normalized scores extracted from the sample Doc_i. A normalized score ns^i_j of a term t^i_j is computed as $ns^i_j = s^i_j / s^i_{\max}$ where $s^i_{\max}$ is the maximal score among all the terms in the bag. A bag of terms T_i is the termhood related to Doc_i if T_i contains only:

- Significant terms – i.e. those scored above the significance threshold ε_s; and
- Valid terms – i.e. those remaining after filtering out the terms that are highly ranked, but have no substantial contribution to the semantics of the domain.

One reasonable way to choose ε_s is to ensure that the terms in the termhood reflect the majority of the stakeholders' opinions. This could be done by taking in those terms from the top of the bag of terms, sorted by term score, having the sum of the scores slightly higher than the 50 per cent of the sum of all scores in the bag of terms.

$Doc = Doc_1, ..., Doc_{n-1}, Doc_n$ is considered *saturated* if:

$$thd(T_{n-1}, T_n) < \varepsilon_{st}, \tag{6}$$

where: *thd* is the termhood difference function computed using the THD algorithm [7] which takes semantically equivalent and orphan terms in consideration; ε_{st} is the saturation threshold chosen empirically by a knowledge engineer for the given domain; T_{n-1}, T_n are the termhoods related to the two final document samples Doc_{n-1}, Doc_n of *Doc*.

It is assumed in our work that the sequence of *thd* values monotonically going down below ε_{st} indicates that Doc_n is a complete document corpus possessing sufficient representativeness. Non-monotonicity of *thd* values sequence signals that the corresponding Δ_{i+1} is either not very relevant to the domain or is a valuable addition containing the terminology not used in the previous samples (Doc_i). Anyhow, saturation indicates that the chosen document corpus is complete.

In order to apply semi-automated ontology mapping technique to compare these extracted requirements and ontology contexts, the requirements have to be represented similarly formally as the ontology. For achieving that:

- Natural language definitions for the terms in the final termhood are collected. An example is given in Figure 2. This activity is performed manually by a knowledge engineer.
- Formalized semantic contexts are built for the terms using the retrieved definitions. This activity could be facilitated by following the OntoElect methodology as described in [7,8].
- The mappings of the constructed semantic contexts to the ontology are created. This activity could be done semi-automatically using the software tools for ontology alignment [11,12].

4.2 Computing Ontology Fitness

For measuring the fitness of the entire ontology or its particular constituents with respect to the domain requirements the OntoElect

methodology [7] recommends to use the metaphor of votes. Votes are computed based on:

- The scores of the respective terms t in R
- The mappings of the terms to the ontology elements

A **mapping** of the term t to ontology O is denoted as the function that establishes a relationship between t and the element of O: $\mu = (t, re, o, cf)$, where re is the relationship type $re \in \{equivalence, membership, subsumption, meronymy, association\}$, o is the element in O, and cf is the confidence factor with a value from $[0, 1]$. Hence, $M_o = \{\mu\}$ is the set of all term mappings to the ontology element o.

A **positive vote** v_o for an ontology element $o \in O$ is denoted as a value reflecting the evidence of referring to o by the term t through the term mapping μ:

$$v_o = \sum_{\mu \in M_o} ns \times w(re) \times cf, \tag{7}$$

where: ns is the normalized score of t; cf is the confidence factor of the respective mapping μ; and $w(re)$ is the weight of the mapping based on the type of the relationship re of μ. The weights are introduced to reflect that different types of mappings could be regarded as the arguments of different strength in favour of this ontological element. Indeed, if a term is *equivalent* to the element, then it is a strong direct argument in favour of the element. However a statement about being an individual member of the element, a direct subsumption of an element, being a part of an element, or having an association to an element is considered as a weaker argument. So the weights are proposed as: *equivalence* – 1.0; *membership* – 0.7; *subsumption*, *meronymy* – 0.5; *association* – 0.3. These values may further be reconsidered if any experimental evidence is collected in this respect. Direct subsumption mappings to very abstract elements in the ontology should however be avoided. For example, all concepts, and therefore the terms categorized as concepts, subsume to the root concept of a **Thing** present in any

OWL ontology. This subsumption mapping has indeed very little to do with domain semantics and therefore should not be counted as an argument for a vote. Valid direct subsumption mappings have to be sought to the most specific possible ontology elements. Indirect subsumption mappings could further be accounted for propagating votes up the concept hierarchy as described below. Propagated votes may be used to further clarify the distribution of the fitness upwards the subsumption hierarchy of the ontology.

So far only direct positive votes with respect to ontology elements have been discussed. So, the overall ontology fitness computed based on these votes reflects only the arguments focused on an element and without any influence on the surrounding of this element. This however might not be fully correct with respect to the fitness of the surrounding elements. Indeed, let us for example assume that the concept of a **Clock** in a Time ontology gets a vote. Then it may be expected that the concept of an **Instrument**, subsuming **Clock** (see also Fig. 2), also qualifies for the part of the value of this vote. A straightforward reason is that, due to the subsumption relationship, the more specific concept inherits the properties of the more abstract concept in the subsumption hierarchy. So the vote has to be propagated up the hierarchy with attenuation – factored empirically or possibly aligned with the proportion of the inherited properties in each individual case.

A **propagated vote** v_o^p for an ontology element $o \in O$ is the value reflecting the contribution of o to the semantics of the ontology element o^{sub} subsumed by o:

$$v_o^p = att \times v_{o^{sub}}, \tag{8}$$

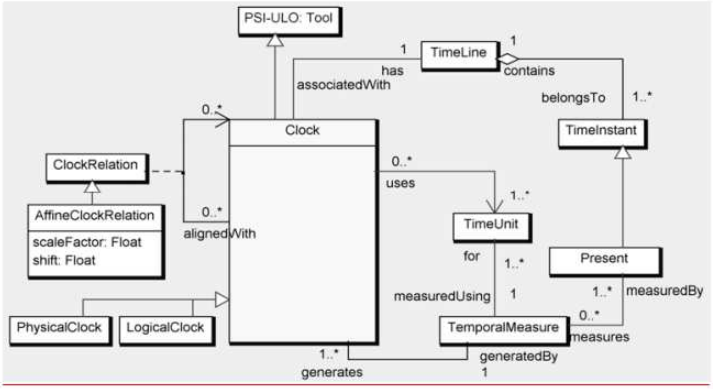
where att is the attenuation coefficient.

Positive and propagated votes provided by the term t are further used for computing the **fitness increments** $\Delta\Phi_o^+$ of the elements in O.

$$\Delta\Phi_o^+ = \sum_{\mu \in M_o} v_o + \sum_{O_o^{sub}} v_o^p, \tag{9}$$

A **Clock** is an **Instrument** to generate the instances of a **TemporalMeasure** of a **TimeInstant** when this **TimeInstant** instance is also the instance of a **Present**. A **Clock** is always associated with a particular single **TimeLine** (there could be **TimeLine**s with no **Clock** but also **TimeLine**s having several different **Clock**s associated with them). Different **Clock**s, associated with the same **TimeLine** or different **TimeLine**s may "run" differently, e.g. quicker or slower and also with offsets compared to each other. Some **Clock**s may be related to each other for enabling a proper comparison of the values they return. This is done by specifying a **ClockRelation**. A specific and most widely used kind of a **ClockRelation** is **AffineClockRelation** which allows aligning different time velocities (using the scaleFactor property) and also time offsets, like delays (using the shift property). A **Clock**, as a measurement instrument, may return a single value (a **TimeStamp** corresponding to a single **TimeUnit**) or several values (the parts of a **TimeStamp** corresponding to different **TimeUnit**s). A **PhysicalClock** and a **LogicalClock** are the two disjoint specializations of a **Clock**.

(a) A natural language definition of the concept of a Clock



(b) A UML class diagram formalizing the natural language definition of a Clock (Figure 2a). The corresponding OWL code for a Clock is omitted.

Figure 2. Term processing pipline by example. The term and semantic context of a Clock.

where O_o^{sub} is the subset of the elements in O which are subsumed by o.

A **negative vote** provided by a term t ($v_t^- = -ns$) is:

- Either a vote based on the term $t \in T^{miss}$ pointing out that t is not described by O. In this case a fitness decrement for the whole ontology O could be computed as:

$$\Delta\Phi_O^- = v_t^- \mid_{t \in T^{miss}} ; \tag{10}$$

- Or a vote pointing out that the term t is in a contradiction with a particular ontology element o. In this case a fitness decrement for the ontology element $o \in O$ could be computed as:

$$\Delta\Phi_o^- = v_t^- . \tag{11}$$

The overall change in ontology fitness caused by the influence of the term t (requirement $r \in R$), being the sum of all positive, propagated, and negative votes could hence be computed as follows:

$$\Delta\Phi_O \mid_t = \sum_O (\Delta\Phi_o^+ + \Delta\Phi_o^-) + \Delta\Phi_O^- . \tag{12}$$

Consequently, the change in overall ontology fitness caused by R is:

$$\Delta\Phi_O \mid_R = \sum_R (\Delta\Phi_O \mid_t), \tag{13}$$

As already mentioned in Section 2.2, all these changes are taking effect if the sphere of influence ns of the requirement $r = (t, ns) \in R$ is more or equal to the distance l between O and D.

4.3 Computing Mappings between Ontologies

The creation of the mappings of the semantic contexts of the terms from the termhood could be done in a partially automated way using

an appropriate ontology matching technique. One possible technique is meaning negotiation using argumentation based on the exchange of presuppositions [10]. This approach has been implemented in several software tools supporting different steps in the mapping generation process:

- Computing the structural changes between two different OWL ontologies and visualizing the difference using an extension to the UML class diagram language could be performed by the Ontology Difference Visualizer tool [11]
- Generation of the mappings between the TBoxes of two different ontologies in the ontology alignment format or as ABox transformation rules could be facilitated using the Structural Difference Discovery Engine [12]

5 Conclusive Remarks

This paper presented the approach to deal with dynamics in knowledge representations, in the form of ontologies, regarding the domains these ontologies are intended to describe. In order to place the reported research in the context of the scientific discipline, the basics of Ontology Engineering, Management, and Change have been concisely presented in Section 2.

The high-level idea followed in the presented work is to understand the dynamics of ontologies in a way similar to the other scientific disciplines – primarily answering the questions about the causes of a change and therefore offering the laws to compute forces, torques and their effect on the motion of ontologies within the domain. Hence, the central part of the presented research deals with an attempt to exploit the analogy with the Newton's law of Universal Gravitation. This law has however to be applied to the objects that do not possess physical mass. Therefore, the proper analogues for a mass, a coordinate grid and its origin; a position, a distance, a motion; and a force (gravitation) have been elaborated – resulting in a theoretical Ontology Gravitation

framework presented in Section 3. This framework is based on the notion and measurement of ontology fitness to the knowledge stakeholder requirements to the description of a particular Domain of Discourse.

It has also been described in Section 4 of the paper that the implementation of the presented theoretical framework is feasible as the supporting techniques, including some software tools already exist. The presentation focused on outlining the opportunities provided by the OntoElect methodology, Ontology Difference Visualizer, and Structural Difference Discovery Engine to help solve the practical problems in:

- Eliciting domain requirements without any direct involvement of the knowledge stakeholders
- Developing structural contexts for multiple word key phrases that indicate the requirements
- Generating the mappings between these structural contexts and the target ontology
- Computing increments and decrements of ontology fitness based on these mappings

The framework presented in the paper has prospects to be applied practically for visualization and analysis of ontology changes in dynamics. The following use cases could be of particular scientific, industrial, and societal value.

Ontology refinement is the implementation of the required changes in an ontology for making it fit the changed stakeholder requirements to the maximal possible extent. In the terms of the Ontology Gravitation framework described above, stakeholder requirements are captured by R for D (Fig. 1), each having also its *ns*. So the changes in these requirements result in the changes to the gravitation field generated by D. These in turn will cause that the ontology O changes its position to reach an equilibrium state in the changed gravitation field of D. This new position of O may appear to be closer to the centre of D's gravitation – which indicates that the changes in the stakeholder requirements

were favourable for the current implementation of O. It may also appear that O will move further out from the gravitation centre of D – indicating that the changes in requirements hint about the necessity to refine O. A simple visualization tool showing the changes in ontology positions in response to the changes in the gravitation field of D may become a powerful instrument for a knowledge engineer to assess and justify the refinement of the particular fragments of the ontology. Such a justification will be based on the acquired knowledge, in a condensed and visualized form, about the causes triggering the needed change.

Anomaly detection in data analytics is about revealing the parts of data that change beyond normal values – hinting about a potential or developing problem in the system that is the source of these data. For example, if a system is a civil community and its environment (D), then it may be producing many diverse streams of observation data coming from various sorts of sensors – like outdoor temperature measurements, water levels, industrial emissions, share prices, cell phone activity, etc. Imagine that each sort of censor measurement is described by its individual ontology which is updated using knowledge extraction from the respective incoming data stream. From the other hand, community requirements R reflect the desire of the stakeholders to live in a comfortable (normal) environment: clean air and water; stable share prices, no traffic hold-ups, etc. If so, it is reasonable to expect that an equilibrium state, involving the abovementioned sensor data ontologies and D, will show how close (normal) or far (abnormal) each sort of sensor measurement is from the normal condition. This visual result may be made available in time sufficient for emergency response to the detected anomaly.

6 Acknowledgements

The theory of gravitation between ontologies and Domains of Discource, based on fitness, has been developed in the SemData project funded by the Marie Cure International Research Staff Exchange Scheme (IRSES) of the 7th Framework Programme of the European Union, grant No PIRSES-GA-2013-612551. OntoElect methodology

has been developed as a part of the PhD project of Olga Tatarintseva. The tools for ontology mapping and alignment have been developed with Anton Copylov and Maxim Davidovsky.

References

[1] I. Newton. *The Principia, Mathematical Principles of Natural Philosophy, a new Translation.*
By I Bernard Cohen and Anne Whitman, preceded by "A Guide to Newton's Principia" by I Bernard Cohen, University of California Press, 1999, ISBN 978-0-520-08816-0, ISBN 978-0-520-08817-7.

[2] Rudi Studer, V. Richard Benjamins, Dieter Fensel. *Knowledge Engineering: Principles and Methods.* In: Data & Knowledge Engineering, 25(1-2):161–197, 1998.

[3] J. Euzenat, P. Shvaiko. *Ontology Matching*, Berlin Heidelberg (DE), Springer-Verlag, 2007.

[4] D. Nardi, R. J. Brachman. *An Introduction to Description Logics.* In The Description Logic Handbook, F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, Eds. Cambridge University Press New York, NY, USA, 2007.

[5] A. Gangemi, C. Catenacci, M. Ciaramita, J. Lehmann. *Modelling ontology evaluation and validation.* In: Sure, Y., Domingue, J. (eds.) ESWC 2006, LNCS 4011, pp. 140–154 (2006).

[6] Fabian Neuhaus, Amanda Vizedom, Ken Baclawski, Mike Bennett, Mike Dean, Michael Denny, Michael Grüninger, Ali Hashemi, Terry Longstreth, Leo Obrst, Steve Ray, Ram Sriram, Todd Schneider, Marcela Vegetti, Matthew West, Peter Yim. *Towards Ontology Evaluation Across the Life Cycle.* In: Applied Ontology. 8 (2013), pp. 179–194, DOI 10.3233/AO-130125.

[7] O. Tatarintseva, V. Ermolayev, B. Keller, W.-E. Matzke. *Quantifying Ontology Fitness in OntoElect Using Saturation- and Vote-*

Based Metrics. In: Ermolayev et al. (eds.) ICT in Education, Research, and Industrial Applications. Revised Selected Papers of ICTERI 2013, CCIS **412**, pp. 136–162, Springer Verlag, Berlin – Heidelberg (2013).

[8] V. Ermolayev, S. Batsakis, N. Keberle, O. Tatarintseva, G. Antoniou. *Ontologies of Time: Review and Trends.* In: Int. J. of Computer Science & Applications, 11(3), pp. 57–115, 2014.

[9] Giorgos Flouris, Dimitris Manakanatas, Haridimos Kondylakis, Dimitris Plexousakis, Grigoris Antoniou. *Ontology Change: Classification and Survey.* In: The Knowledge Engineering Review, 23(2), pp. 117-152, 2008. doi:10.1017/S0269888908001367.

[10] V. Ermolayev, N. Keberle, W.-E. Matzke, V. Vladimirov. *A Strategy for Automated Meaning Negotiation in Distributed Information Retrieval.* In: Y. Gil et al. (Eds.): ISWC 2005 Proc. 4th Int. Semantic Web Conference (ISWC'05), 6-10 November, Galway, Ireland, LNCS 3729, pp. 201–215 (2005).

[11] V. Ermolayev, A. Copylov, N. Keberle, E. Jentzsch, W.-E. Matzke. *Using Contexts in Ontology Structural Change Analysis.* In: Ermolayev, V., Gomez-Perez, J.-M., Haase, P., Warren, P, (eds.) CIAO 2010, CEUR-WS, vol. 626 (2010).

[12] M. Davidovsky, V. Ermolayev, V. Tolok. *Agent-Based Implementation for the Discovery of Structural Difference in OWL-DL Ontologies.* In: H.C. Mayr et al. (Eds.): UNISCON 2012, LNBIP 137, pp. 87–95 (2013).

Vadim Ermolayev

Received July 1, 2015

Department of IT, Zaporizhzhya National University,
66 Zhukovskogo st., 69063, Zaporizhzhya, Ukraine
E–mail: vadim@ermolayev.com

Part 8

Natural language processing

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Word formation problems in Romanian and their solving by P systems

Artiom Alhazov Svetlana Cojocaru
Constantin Ciubotaru Alexandru Colesnicov
Ludmila Malahov Mircea Petic

Abstract

This paper is an overview of authors' results in solving word formation problem for Romanian. Generalized formal models for inflexion and affixation are discussed. Two different approaches of their application are presented: a classical one based on specific grammars, and a bio-molecular one using membrane systems.

1 Introduction

Word formation is a well-known problem in natural language morphology. In high inflectional languages including Romanian, we found extensive patterns for word inflection (declension of nouns and adjectives, conjugation of verbs). Word derivation with prefixes and suffixes is even more complicated. Inflection keeps word meaning while affixation modifies it. Consequently, the semantic checking of the derived word became necessary. Whereas these two processes seem to be different, they have enough in common to be described by a unified formal model.

In this paper, we will present one such model of word formation. It is described in Sec. 2.

Several approaches are possible to exploit this model. We will discuss two of them: classical (Sec. 3), and bio-molecular using membrane systems, or P systems. The latter approach is characterized by its intrinsic massive parallelism. Sec. 4–7 discuss this. P systems can only

be simulated [1]. Nevertheless, this approach is not pure theoretical as its non-standard patterns of parallelization may be used on existing supercomputers.

2 Word Formation Model

Let us consider a finite alphabet Σ. Below, all words are elements in Σ^*, and all languages are over Σ (subsets of Σ^*). We will also permit marks on symbols of words from Σ^*.

We will present possible root alternations as a finite set A of ordered pairs of words. Four finite languages Pref, RR, Suf, and T present prefixes, roots, suffixes, and terminations. T may contain the empty word.

We will sometimes write elements of A in the form $x \to y$ focusing on the operation of root alternation. To present operations of adding a prefix or a suffix, we will use *markings* $\overline{\text{Pref}}$ or $\widehat{\text{Suf}}$. This denotes a set Pref or Suf, where all symbols of each word have lines or hats over them. We denote the marked terminations by $\boxed{\text{T}} = \{\boxed{\text{t}} \mid t \in T\}$, and the termination rewriting rules by $\overset{\square}{T} = \{\boxed{t_1} \to \boxed{t_2} \mid t_1, t_2 \in T\}$.

The set of operations is $Op = \overline{\text{Pref}} \cup \widehat{\text{Suf}} \cup \overset{\square}{T} \cup A$ (adding a prefix, or adding a suffix, or adding a termination, or performing root alternation).

Now let us take a finite language M over Op. A word formation step corresponding to a control word $s = op_{i_1} \cdots op_{i_k} \in M$ consists of k operations from the set Op described above. Let us denote by $op(w)$ the result of operation op over a word w. Let us mention that the result of some operations over some word may be undefined.

- $\overline{p}(w) = \overline{p}w$,
- $\widehat{s}(w\boxed{\text{t}}) = w\widehat{s}\boxed{\text{t}}$,
- $(\boxed{\text{x}} \to \boxed{\text{y}})(w\boxed{\text{x}}) = w\boxed{\text{y}}$,
- $(x \to y)(w_1 x w_2) = w_1 y w_2$,
- $(o_{i_1} \cdots o_{i_k})(w) = o_{i_1}(\cdots(o_{i_k}(w))\cdots)$.

M *accepts* a language L_0 if:

- L_0 is obtained by removing the prefix, suffix and termination marks from the words of some language L;
- if $w \in \mathrm{RR}$, then $w\boxed{\mathrm{t}} \in L$;
- if $w \in L$ and $s \in M$, then $s(w) \in L$ is defined;
- L is a minimal language with the said properties.

We would prefer the acceptor M to be supplemented by the lexical decomposition of the input L_0 in the form of the correspondence between operation steps $s \in M^*$, the words that were produced by s, and the corresponding arguments for s. We will show the examples below.

3 Classical Approach

To inflect and to derivate Romanian words we developed a special grammar formalizing word formation.

Inflexion. We use the paradigmatic approach for inflexion model. Here paradigms and sub-paradigms are sets of terminations enlisted into the indexed list L: $N_i = \{t_1, t_2, \ldots, t_{n_i}\} \in T$. We use two operations from those listed above: adding a termination, and performing root alternation.

Let us consider a scattered context grammar [2] rule:

$$[/]^*[\#][N_1]a_1\neg b_1 a_2 \ldots a_{n-1}\neg b_{n-1}a_n \to a'_1\neg b_1 a'_2 \ldots a'_{n-1}\neg b_{n-1}a'_n[N_2],$$

where a_i, a'_i are arbitrary words and either b_i is a nonempty word, or the special symbol $*$ stands instead of b_i. N_1, N_2 are the termination set indexes. The interpretation of this rule is as follows.

Let w be a word-lemma. Every sign / if any indicates cutting the last letter from w. Word v obtained after the deletions is considered as a root if N_1 is not empty, and N_1 is its index in the termination set list L. In any case, the word v should have the form

$$f_0 a_1 f_1 a_2 f_2 \ldots a_{n-1} f_{n-1} a_n f_n,$$

where each f_i is an arbitrary (possible empty) word, not containing (for $i = 1, 2, \ldots, n-1$) the veto subword b_i. If there exists more than one representation of this kind, the first one should be selected scanning v

from the left to the right, or vice versa if the sign # is present. The special character $*$ instead of b_i admits arbitrary f_i.

In this context, the parallel substitution

$$a_1, a_2, \ldots, a_n \to a'_1, a'_2, \ldots, a'_n,$$

is produced, generating a new root $v' = f'_0 a'_1 f'_1 a'_2 f'_2 \ldots a'_{n-1} f'_{n-1} a'_n f'_n$ and N_2 is its termination set index in L.

So, in order to generate word-forms it is sufficient to interpret the corresponding grammar rules [3, 4].

To apply this method of inflexion, we should know the morphological group of the given word. See discussion and a partial solution in [3]. (An algorithm to determine the morphological group of an arbitrary Romanian word using P systems can be found in [5].)

We developed 866 grammar rules and 320 ending sets that are sufficient to formalize the inflexion process of productive parts of speech for the Romanian language.

Obviously, for each ending we can establish a correspondence with morphological characteristics of the word-form, thus obtaining a possibility of morphological annotation.

Derivation. The particularities of the derivational morphology mechanisms help in lexical resources extension without any semantic information. The developed approaches and mechanisms were applied to Romanian but they can be applicable to other languages [6].

Let Σ be an alphabet of natural language, V — set of vowels, C – set of consonants, $r \in$ RR — the root/stem, $p \in$ Pref – prefix, $s \in T \setminus$ Suf – suffix, brackets [ ] indicate that the included morphem in the paranthesis can be missing from the word structure. From the analysis of the process of derivation we can infer the following rules of modifications of derivatives:

1. $[p]r = [p]r'v \to [p]r's$, where $v \in V$, for example, răzbuna $\to$ răzbun (a)ător, corresponds to $r =$ răzbuna, $r' =$ răzbun, $v' =$ a, $s =$ {ator}.

2. $[p]rs = [p]rls' \to [p]rs'$, where $l \in L$, for example, nărui $\to$ nărui(e)ală, corresponds to $r =$ nărui, $s =$ eală, $l =$ e, $s' =$ ală.

3. $[p]rs = [p]rl_1l_2s' \to [p]rs'$, where $l_1, l_2 \in L$, for example, încleia $\to$ înclei(ea)ală, corresponds to $r =$ încleia, $l_1 =$ e, $l_2 =$ a, $s =$ eală and $s' =$ lă.

In general, the process of derivation can be presented as a set of rules of the form:

$$\alpha_1\beta_1\alpha_2\beta_2 \ldots \alpha_n\beta_n \to [p]\alpha_1\beta_1'\alpha_2\beta_2' \ldots \alpha_n\beta\prime_n[s],$$

where $|\alpha_i| \geq 0$, $|\beta_i| \geq 0$, $|\beta_i'| \geq 0$.

For example: dărima $\to$ dărimătură, the alternation a-ă; ceaţă $\to$ ceţos, the alternation ea-e; contagios $\to$ contagioasă, the alternation io-ioa.

Studies in derivation process allow us to conclude that we cannot propose an effective general algorithm for automatic derivation but we can highlight some models of derivation, for which construction of such algorithms is possible. We will underline below the four most important models of derivation.

Affixes substitution. Let x_1 be a word of the form $x_1 = \alpha_1\omega$, where α_1 is a prefix. After the substitution $\alpha_1 \to \alpha_2$ we obtain the word $x_2 = \alpha_2\omega$, where x_2 is the obtained derivative, for example, *în*chide-*des*chide. The same for the derivatives $x_1 = \omega\beta_1$, with the suffixe β_1. After the substitution $\beta_1 \to \beta_2$ we obtain the word $x_2 = \omega\beta_2$, for example, corig*enţă*-corig*ent*.

Derivatives projection. Let ω be a word, α – a prefix, β – a suffix, then the following relations are valuable:

$(\omega \to \alpha\omega)\bigwedge(\omega \to \omega\beta) \Rightarrow (\omega \to \alpha\omega\beta)$, for example, (a lucra $\to$ lucr(a)ă*tor*)$\bigwedge$ (a lucra $\to$ a *pre*lucra) $\Rightarrow$ (a lucra $\to$ *pre*lucr(a)ă*tor*);

$(\omega \to \alpha\omega)\bigwedge(\omega \to \alpha\omega\beta) \Rightarrow (\omega \to \omega\beta)$, for example, (a capitula $\to$ capitula*ţie*) $\bigwedge$(a capitula $\to$ *re*capitula*ţie*) $\Rightarrow$ (a capitula $\to$ capitula*ţie*);

$(\omega \to \alpha\omega\beta)\bigwedge(\omega \to \omega\beta) \Rightarrow (\omega \to \alpha\omega)$, for example, (a centraliza $\to$ *des*centraliza*tor*)$\bigwedge$(a centraliza $\to$ centraliza*tor*) $\Rightarrow$ (a centraliza $\to$ *des*centraliza);

Formal derivational rules. For example, let an adjective be of the form $\omega\prime = \omega\beta$, where $\beta \in$ {-tor, -bil, -os, -at, -it, -ut,-ind, -înd},

as a result we will obtain the derivatives of the form $\omega'' = ne\omega\beta$, for example, invidios $\rightarrow$ *ne*invidios.

Derivational constraints. These can be functions of the form:

$$f : \{wrd, pos, mod, sla, fgw, mvca \dots\} \rightarrow derivative$$

where *wrd* is a word to derivate, *pos* − part of speech of *wrd*, *mod* − model of derivation, *sla* − the set of letters to which the affix is attached, *fgw* − flection group of *wrd*, *mvca* − modifications and vocalic or consonant alternations. For example, f : { a spinteca, verb, des< *verb* >, ..., V14, double consonant avoiding } $\rightarrow$ de(s)spinteca.

As derivatives generation represents an overgenerating mechanism, these derivational models implementation needs a level of validation [6], in order to enrich existent digital lexicons.

Generation of derivatives is not a trivial problem, because the process does not have a regular mechanism. Solution to store all derivatives of a dictionary is a reasonable one, because these derivatives still will not cover the full diversity of language, being in continuous evolution [7].

4 P systems with String Objects and Input

A membrane system, or P system, consists of nested *membranes* whose *regions* contain multisets of *objects*. Objects evolve in a synchronous and maximally parallel manner according to given *evolution rules* associated with membranes. See [8] for further details.

Let O be a finite set of elements called symbols, then the set of words over O is denoted by O^*, and the empty word is denoted by λ.

A transitional P system with string-objects and input is a tuple

$$\Pi = \big(O, \Sigma, \mu, M_1, \cdots, M_p, R_1, \cdots, R_p, i_0\big),$$

where:

- O is the working alphabet of the system whose elements are called objects.
- $\Sigma \subset O$ is an input alphabet.

- μ is a membrane structure (a rooted tree) consisting of p membranes.
- M_i is an *initial* multiset of strings over O (see an exception below) in region i, $1 \leq i \leq p$.
- R_i is a finite set of rules defining the behavior of objects from O^* in region i, $1 \leq i \leq p$, as described below.
- i_0 identifies the input region. M_{i_0} is restricted to the input alphabet Σ.

For our current purposes, the membrane structure is supposed to be unchanged during computation.

In this paper we consider string rewriting with target indications. A rule $x \to (y, tar) \in R_i$ can be applied to a string uxv in region i, resulting in a string uyv in region specified by $tar \in \{in_j \mid 1 \leq j \leq p\} \cup \{here, out\}$. Rule $x \to (y, here)$ can be shortened to $x \to y$.

We can also write rules of the type $a \to (u_1, tar_1)||(u_2, tar_2)|| \cdots ||(u_k, tar_k)$ that transform any string of the form $w_1 a w_2$ into the multiset of strings $w_1 u_1 w_2$, $w_1 u_2 w_2$, $\cdots$, $w_1 u_k w_2$ and send the results in the corresponding region.

We assume the following **computation mode**: whenever there are multiple ways to apply different rules or the same rule to a string, all possible results are produced. Each possible result is obtained from a different copy of the string that is replicated, or assumed to be present in sufficient number of copies to allow this.

The computation ends when no rule is applicable, and the strings sent *out* form the result of the computation.

5 Inflection by P Systems

Given the set of word forms, an opened productive class assumes that these words are identified by their inflection group [9]. The inflection group is characterized by the set $G = \{\alpha, R_G, F_G\}$, where $|\alpha| \geq 0$. α is the ending that is reduced during inflection, F_G is the set of the lists of flectives presenting the inflexion paradigm, R_G is the set of the rules, which indicate vowel/consonant alternation of type $a \to u$, $a \in \Sigma^+$, $u \in \Sigma^*$. A separate membrane system Π_G is developed for

each inflexion group G.

We will investigate two cases: *without* alternations, and *with* vowel and/or consonant alternation.

No alternation model. For any inflection group without alternation $G = (\alpha, \emptyset, \{f_{1_G}, f_{2_G}, \cdots, f_{n_G}\})$:

$$\begin{aligned} \Pi_G &= (O, \Sigma, [\ ]_1, \emptyset, R_1, 1), \text{ where} \\ O &= \Sigma = \{\mathrm{a}, \cdots, \mathrm{z}, \#\} \text{ (extended Romanian alphabet)}, \\ R_1 &= \{\alpha\# \to (f_{1_G}, out)||(f_{2_G}, out)||\cdots||(f_{n_G}, out)\} \end{aligned}$$

If this system receives an input words $w'\alpha\#$ belonging to the inflection group G, it sends all its word forms out of the system in one step. Clearly, Π_G is non-cooperative if $\alpha = \lambda$, but non-cooperativeness is too restrictive in general, since the system would not be able in this case to distinguish the termination to be reduced from any other occurrence of α.

Model with alternations. Let G be an arbitrary inflection group, with $m-1$ alternations $a_1 = a_1^{(1)}a_2^{(1)}\cdots a_{n_1}^{(1)}, \cdots, a_m = a_1^{(m)}a_2^{(m)}\cdots a_{n_m}^{(m)}$. Let the set of flectives consist of s subsets, and for subset $F_{k_G} = \{f_1^{(k)}, \cdots, f_{p_1}^{(k)}\}$, $1 \le k \le s$, the following alternations occur: $a_1 \to u_1^{(k)}$, $\cdots$, $a_m \to u_m^{(k)}$ (the alternations are fictive for $k = 1$), and $\bigcup_{k=1}^{s} F_{k_G}$ corresponds to a complete paradigm. For instance, Example 2 corresponds to $s = 2$ sublists (singular and plural), and $m - 1 = 2$ alternations.

The associated P system should perform the computation

$$w\# = \prod_{j=1}^{m-1} (w_j a_j)\, w_m \alpha\# \Rightarrow^*$$

$$\Rightarrow^* \left\{ \prod_{j=1}^{m-1} \left(w_j u_j^{(k)}\right) w_m f_{i_k} \mid 1 \le k \le s,\ f_{i_k} \in F^{(k)} \right\},$$

where $u_j^{(1)} = a_j$, $1 \le j \le m$.

The system processes the first alternation at its leftmost occurrence, the second alternation at its leftmost occurrence which is to the right of the first one, etc. More formally, our P system discovers the representation of the input string as $\prod_{j=1}^{m-1} (w_j a_j)\, w_m \alpha$, where a_j has no other occurrences inside $w_j a_j$ except as a suffix.

Theoretically, overlapping occurrences or occurrences with context can be handled by rules with a longer left-hand side. A different order of occurrences of the alternations can be handled by renumbering the alternations. Should the specification of a group require, e.g., second-leftmost occurrence for $a \to u$, this can be handled by inserting a fictive substitution $a \to a$ before $a \to u$, etc.

We construct the following P system, which takes the input in the form

$$\#_l w \#_r = \#_l \prod_{j=1}^{m-1} (w_j a_j)\, w_m \alpha \#_r.$$

$$\begin{aligned}
\Pi''_G &= (O, \Sigma, [\ ]_1, \emptyset, R_1, 1), \text{ where} \\
\Sigma &= V \cup \{\#_l, \#_r\}, \\
O &= \Sigma \cup E, \\
E &= \{A_{k,j} \mid 1 \le k \le s,\ 0 \le j \le m\}, \\
V &= \{a, \cdots, z\} \text{ is the Romanian alphabet,}
\end{aligned}$$

and the rules are as follows:

$$\begin{aligned}
R &= \{\#_l \to A_{1,0} || \cdots || A_{s,0}\} && (1) \\
&\cup \{A_{k,j-1}\gamma \to \gamma A_{k,j-1} \mid \gamma \in V \setminus \{a_1^{(j)}\}, \\
&\quad 1 \le k \le s,\ 1 \le j \le m\} && (2) \\
&\cup \{A_{k,j-1} a_1^{(j)} v\gamma \to a_1^{(j)} A_{k,j-1} v\gamma \mid a_1^{(j)} v \in Pref(a_j), \\
&\quad |v| < |a_j| - 1, \gamma \in V \setminus \{a_1^{(|v|+2)}\},\ 1 \le k \le s, 1 \le j \le m\} && (3) \\
&\cup \{A_{k,j-1} a_j \to u_j^{(k)} A_{k,j} \mid 1 \le k \le s,\ 1 \le j \le m\} && (4) \\
&\cup \{\alpha A_{k,m} \#_r \to (f_1^{(k)}, out) || \cdots || (f_{p_m}^{(k)}, out) \mid 1 \le k \le s\}. && (5)
\end{aligned}$$

The rules are presented as a union of 5 sets. The rule in the first set replicates the input to perform different inflections. The symbol $A_{k,j}$ is a marker that will move through the string. Its index k corresponds to the inflection subset, while index j tells how many alternations have been carried out so far.

The rules in the second set allow the marker to skip a letter if it does not match the first letter needed for the current alternation.

The rules in the third set allow the marker to skip one letter if some prefix of the needed subword is found, followed by a mismatch.

The rules in the fourth set carry out an alternation.

The last set of rules perform the replicative substitution of the flectives.

This system halts in at most $|w| + 2$ steps.

6 Derivation and Parsing by P Systems

We proceed with parsing as the reverse process of the generation. For each possible decomposition of the string, the system sends outside a string, obtained from the input by erasing the endmarkers and inserting hyphens (for technical reasons, letters in prefixes and suffixes are marked, the reverse alternations are performed in both the termination and the rest of the word, and the termination is moved to the left of suffixes). In the notation below, we use $'$ as a morphism: u' is a string obtained from u by priming all its letters.

We construct a P system that accepts words x provided by the beginning and ending markers: $\$_1 x \$_2$. We use an enumeration of elements of Op and T: $Op = \{o_1, \cdots, o_k\}$ and $T = \{t_1, \cdots, t_n\}$. We recall that elements o_j, $1 \leq j \leq k$ are of the following forms: $\overline{p}$, $\widehat{s}$, $\boxed{t_1} \to \boxed{t_2}$, and $x \to y$, where $p \in \text{Pref}$, $s \in \text{Suf}$, $t_1, t_2 \in T$ and $x, y \in V^*$. We also define a set $W = Suf(M^r)$ of suffixes of the mirror language of M; words from W may appear in angular brackets. This corresponds to operations remaining to be undone at possible points of the parsing process.

We suppose that the alternations are only allowed in the root of the word, not in the prefixes or suffixes to be removed. We proceed as

follows: all reverse alternations are replaced with the prime version of the letters. Once the choice is made to stop performing the operations (the string is in a region corresponding to its termination, and the control symbol is removed), every letter can be unprimed, and then the result is sent out if some word from RR is obtained between the markers.

$$
\begin{aligned}
\Pi &= (O, \mu, \Sigma, w_1, w_2, w_{o_1}, \cdots, w_{o_k}, w_{t_1}, \cdots, w_{t_n}, \\
&\quad R_1, R_2, R_{o_1}, \cdots, R_{o_k}, R_{t_1}, \cdots, R_{t_n}, i_0 = 2), \\
O &= V \cup V' \cup \overline{V} \cup \widehat{V} \cup \boxed{\mathrm{T}} \cup Op \cup \{\$_1, \$_2, -\},\ V = \{a, \cdots, z\}, \\
V' &= \{a', \cdots, z'\},\ \overline{V} = \{\overline{a}, \cdots, \overline{z}\},\ \widehat{V} = \{\widehat{a}, \cdots, \widehat{z}\},\ \Sigma = V \cup \{\$_1, \$_2\}, \\
\mu &= [\ [\ ]_2[\ ]_{o_1} \cdots [\ ]_{o_k}[\ ]_{t_1} \cdots [\ ]_{t_n}\ ]_1, \\
w_i &= \lambda,\ i \in \{1, 2\} \cup Op \cup T, \\
R_1 &= \{\langle\rangle \to \langle w \rangle \mid w \in M^r\} \cup \{\langle o \to (\langle, in_o) \mid o \in Op\}, \\
&\cup\ \{\langle\rangle \boxed{\mathrm{t}} \to (\lambda, in_t) \mid t \in T\}, \\
&\cup\ \{\$_1 q - t\$_2 \to (q - t, out) \mid qt \in \mathrm{RR},\ t \in T\}, \\
R_2 &= \{t\$_2 \to (\$_2\langle\rangle \boxed{\mathrm{t}}, out) \mid t \in T\}, \\
R_{\overline{p}} &= \{\$_1 p \to (\overline{p} - \$_1, out)\},\ p \in \mathrm{Pref}, \\
R_{\widehat{s}} &= \{s\$_2 \to (\$_2 - \widehat{s}, out)\},\ s \in \mathrm{Suf}, \\
R_q &= \{\boxed{t_2} \to (\boxed{t_1}, out)\},\ q = \boxed{t_1} \to \boxed{t_2},\ t_1, t_2 \in T. \\
R_a &= \{(z \to x', out) \mid'^{-1} (z) = y\},\ a = (x \to y) \in A, \\
R_t &= \{a' \to a \mid a \in V\} \cup \{\$_2 \to (-t\$_2, out)\},\ t \in T.
\end{aligned}
$$

The notation $'^{-1}$ means removing all primes from the letters of the argument. Although it assumes an exponential number of rules with respect to the size of a root alternation, this size is never too long.

7 Examples of Parsing Romanian Words

We start by illustrating the work of the last P system by an example of a fragment of a computation, where Pref = {des}, RR = {praf},

Suf = {ui,re}, $T = \{\lambda\}$ and $M = \{(\overline{d}\ \overline{e}\ \overline{s}), (a \to \breve{a})(\widehat{ui}), (\widehat{re})\}$, and the system processes input \$$_1$despărfuire\$$_2$. For conciseness, we only list the first evolution of the copies of the string leading to the output, using the notation (string,region). (The other two are obtained if the prefix des is marked before both suffixes or after one of them, yielding the same results, while for technical reasons some strings remain blocked in the system, not contributing to the result).

$$(\$_1\text{despărfuire}\$_2, 2) \Rightarrow (\$_1\text{despărfuire}\$_2\langle\rangle\boxed{\lambda}, 1) \Rightarrow$$
$$(\$_1\text{despărfuire}\$_2\langle(\widehat{re})\rangle\boxed{\lambda}, 1) \Rightarrow (\$_1\text{despărfuire}\$_2\langle\rangle\boxed{\lambda}, (\widehat{re})) \Rightarrow$$
$$(\$_1\text{despărfui}\$_2 - \widehat{re}\langle\rangle\boxed{\lambda}, 1) \Rightarrow (\$_1\text{despărfui}\$_2 - \widehat{re}\langle(\widehat{ui})(a \to \breve{a})\rangle\boxed{\lambda}, 1)$$
$$\Rightarrow$$
$$(\$_1\text{despărfui}\$_2 - \widehat{re}\langle(a \to \breve{a})\rangle\boxed{\lambda}, (\widehat{ui}) \Rightarrow$$
$$(\$_1\text{despărf}\$_2 - \widehat{ui} - \widehat{re}\langle(a \to \breve{a})\rangle\boxed{\lambda}, 1)$$
$$\Rightarrow (\$_1\text{despărf}\$_2 - \widehat{ui} - \widehat{re}\langle\rangle\boxed{\lambda}, (a \to \breve{a}) \Rightarrow$$
$$(\$_1\text{despra}'\text{f}\$_2 - \widehat{ui} - \widehat{re}\langle\rangle\boxed{\lambda}, 1)$$
$$\Rightarrow (\$_1\text{despra}'\text{f}\$_2 - \widehat{ui} - \widehat{re}\langle(\overline{d}\ \overline{e}\ \overline{s})\rangle\boxed{\lambda}, 1) \Rightarrow$$
$$(\$_1\text{despra}'\text{f}\$_2 - \widehat{ui} - \widehat{re}\langle\rangle\boxed{\lambda}, (\overline{d}\ \overline{e}\ \overline{s})) \Rightarrow$$
$$(\overline{d}\ \overline{e}\ \overline{s} - \$_1\text{pra}'\text{f}\$_2 - \widehat{ui} - \widehat{re}\langle\rangle\boxed{\lambda}, 1)$$
$$\Rightarrow (\overline{d}\ \overline{e}\ \overline{s} - \$_1\text{pra}'\text{f}\$_2 - \widehat{ui} - \widehat{re}, \lambda) \Rightarrow (\overline{d}\ \overline{e}\ \overline{s} - \$_1\text{praf}\$_2 - \widehat{ui} - \widehat{re}, \lambda) \Rightarrow$$
$$(\overline{d}\ \overline{e}\ \overline{s} - \$_1\text{praf-}\$_2 - \widehat{ui} - \widehat{re}, 1) \Rightarrow (\overline{d}\ \overline{e}\ \overline{s} - \text{praf- } -\widehat{ui} - \widehat{re}, 0).$$

By inspecting the examples, we have come to the conclusion that a derivation step can include, in the worst case, a prefix, a suffix, two root alternations and replacing a termination with another one. Some of the above mentioned operations may be absent.

We should note that the division of a word into morphemes may sometimes differ from the one commonly accepted in linguistics. However, this should not restrict the generality of the approach, and we did so in order to simplify the explanation.

In the parsing process described above, we accounted for the prefixes, suffixes, root alternations and the terminations. Romanian language has 86 prefixes and approx. 600 suffixes [7]. Processing the dictionary [10] (not claiming its comprehensiveness) let us distinguish the following types of root alternations during the word derivation:

- of vowels: a→ ă, a→ e, e→ ă, o→ u, î→ i, ă→ e, ea→ e, e→ ea, oa→ o, oa→ u, ia→ ie
- of consonants: t→ ţ, d→ z, h→ ş, z→ j, d→ j, t→ c, t→ ci

Note that, if desired, we can use the context information to refine the scope of the the root alternation rules, e.g., if we only wanted to perform the alternation a→ ă between letters t and r, we could write this as a rule tar→ tăr, which does not affect the model at all.

The set of terminations that we use in our algorithm (set T) consists of terminations for nouns and adjectives (ă, e, ea, a, i, ică, the empty termination λ, u, o, a, l, iu, ui, iu, ie, uie) and those for verbs (a, ea, e, i, î).

We now proceed with some more examples of input and output, so let us agree that

$$
\begin{aligned}
\text{Pref} &\supseteq \{\text{im,în,de,re,des}\},\\
\text{RR} &\supseteq \{\text{pune,flori,flex,scrie,cicl,fac,tânăr,fată,mult,deştept,brad,praf}\},\\
\text{Suf} &\supseteq \{\text{ere, are, ibil, re, ire, i, iţ, im, uţ, ui}\},\\
T &\supseteq \{\lambda, \text{ă}, \text{e}\},\\
A &\supseteq \{\text{â} \to \text{i}, \text{ă} \to \text{e}, \text{a} \to \text{e}, \text{t} \to \text{ţ}, \text{e} \to \text{ea}, \text{a} \to \text{ă}\},\\
M &\supseteq \{(\overline{\text{î}}\ \overline{m}), (\overline{i}\ \overline{n}), (\overline{d}\ \overline{e}), (\overline{r}\ \overline{e}), (\overline{d}\ \overline{e}\ \overline{s}), (\widehat{e\,r\,e}), (\widehat{a\,r\,e}), (\widehat{i\,b\,i\,l}), (\widehat{r\,e}), (\widehat{i\,r\,e}),\\
&\quad (\text{â} \to \text{i})(\text{ă} \to \text{e})(\overline{\text{î}n})(\widehat{i}),\ (\text{a} \to \text{e})(\widehat{i\text{ţ}}),\ (\text{t} \to \text{ţ})(\widehat{i\,m}), (\boxed{\lambda} \to \boxed{\text{e}})\\
&\quad (\text{e} \to \text{ea})(\boxed{\lambda} \to \boxed{\text{ă}}),\ (\text{a} \to \text{ă})(\widehat{u\text{ţ}}),\ (\text{a} \to \text{ă})(\widehat{u\,i})\}.
\end{aligned}
$$

Examples without root alternations: words \$$_1$înflorire\$$_2$, \$$_1$flexibil\$$_2$, \$$_1$descriere\$$_2$, \$$_1$reciclare\$$_2$, \$$_1$desfacere\$$_2$ (burst into blossom, flexible, description, recycling, disassembling) will yield output $\overline{\text{î}}\ \overline{n}$-flori--$\widehat{r\,e}$, flex--$\widehat{i\,b\,i\,l}$, $\overline{d}\ \overline{e}$-scrie--$\widehat{r\,e}$, $\overline{r}\ \overline{e}$-cicl--$\widehat{a\,r\,e}$, and $\overline{d}\ \overline{e}\ \overline{s}$-fac-$\widehat{e\,r\,e}$, respectively.

Examples with root alternations: words \$$_1$întineri\$$_2$, \$$_1$fetiţă\$$_2$, \$$_1$mulţime\$$_2$, \$$_1$deşteaptă\$$_2$, \$$_1$brăduţ\$$_2$ and \$$_1$desprăfuire\$$_2$ (youthen, little girl, multitude, dignified (fem.), small spruce, undusting) will yield output $\overline{\text{î}n}$-tânăr--$\widehat{i}$, fat-ă-$\widehat{i\text{ţ}}$, mulţ--$\widehat{i\,m}$, deştept-, brad-$\widehat{u\text{ţ}}$, and $\overline{d}\ \overline{e}\ \overline{s}$-praf--$\widehat{u\,i} - \widehat{r\,e}$, respectively.

8 Conclusions

The paper discussed classical and P systems approaches used to word formation in Romanian, namely, inflection and affixation of nouns, adjectives, and verbs as most productive parts of speech.

The particularities of the derivational morphology mechanisms help in lexical resources extension without semantic information. Studies on derivation process allow us to conclude that we cannot propose an effective algorithm for automatic derivation in general, but we can highlight some models of derivation, for which construction of such algorithms is possible.

In particular, we described a membrane parsing model taking into account alternations in the root dependent of fixed prefixes and suffixes. We may deduce that these models can be used not only for Romanian but for other languages. These models can also be integrated into another NLP applications to solve more complicated problems in computer linguistics.

Acknowledgement

This work was partially supported by NATO.NUKR.SFPP 984877 "Modeling and Mitigation of Social Disasters Caused by Catastrophes and Terrorism".

References

[1] V. Macari, G. Magariu, T. Verlan. *Simulator of P-Systems with String Replication Developed in Framework of P-Lingua 2.1.* Computer Science Journal of Moldova, v. **18**, Nr. 2(53), 2010, pp. 246–268. – ISSN 1561–4042.

[2] Sh. Greibach, J. Hopcroft. *Scattered context grammars.* Journal of Computer and System Sciences. V. **3**, Issue 3, August 1969, pp. 233–247.

[3] S. Cojocaru. *The ascertainment of the inflexion models for Romanian.* Computer Science Journal of Moldova. **14**, 1(40), 2006, pp. 103–112.

[4] E. Boian, C. Ciubotaru, S. Cojocaru, A. Colesnicov, L. Malahov, M. Petic. *Creation and Development of the Romanian Lexical Resources.* In: Proceedings of Recent Advances in Natural Language Processing, Hissar, Bulgaria, September 11–14, 2011, pp. 678–685

[5] S. Cojocaru, E. Boian. *Determination of inflexional group using P systems.* Computer Science Journal of Moldova. 2010, 18(1), pp. 70–81. ISSN 1561–4042.

[6] S. Cojocaru, E. Boian, M. Petic. *Stages in Automatic Derivational Morphology Processing.* In: Knowledge Engineering, Principles and Techniques, KEPT2009, Cluj-Napoca University Press, 2009, pp. 97–104.

[7] M. Petic. *Generative mechanisms of Romanian derivational morphology.* In: Memoirs of the Scientific Section of the Romanian Academy, Series IV, Tome XXXIV, Bucureşti: Editura Academiei, 2011, pp. 21–30.

[8] Gh. Păun, G. Rozenberg, A. Salomaa. *The Oxford Handbook of Membrane Computing*, Oxford University Press, 2010, pp. 118–143.

[9] A. Lombard, C. Gâdei. *Morphological Romanian Dictionary (Dictionnaire morphologique de la langue roumaine).* Bucureşti, Editura Academiei, 1981. – In French.

[10] S. Constantinescu. *Dictionary of Derived Words*, HERRA, Bucharest, 2008. 288 pp. – In Romanian.

Artiom Alhazov, Constantin Ciubotaru,
Svetlana Cojocaru, Alexandru Colesnicov,
Ludmila Malahov, Mircea Petic

Received July 10, 2015

Institute of Mathematics and Computer Science
Str. Academiei 5, Chişinău, MD-2028, Moldova
Phone: +373 22 72 59 82
E–mails: {artiom.alhazov,constantin.ciubotaru,svetlana.cojocaru,kae,mal,mircea.petic}@math.md

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Experiments on cross-language identity resolution*

Zinaida Apanovich, Alexander Marchuk

Abstract

This paper describes approaches to the cross-language identity resolution problem that arises when the English datasets are used to populate the content of Russian scholarly knowledge bases. The goal is to associate personal entities with Russian names, described in the Open Archive of the Russian Academy of Sciences, with their publications written in English. A new approach combining attribute-based identity resolution with the textual analysis of publications attributed to these entities has been proposed. The dataset of the Open Archive of the Russian Academy of Sciences and digital library SpringerLink are used as test examples.

Keywords: Linked Open Data, cross-language identity resolution, tf-idf, LDA, Jaro-Winkler

1 Introduction

Name ambiguity in the context of bibliographic citation records is a difficult problem that affects the quality of content in digital libraries. The library community has been working on it for a long time [1, 2]. In the context of Linked Open Data with the increasing data traffic on a global scale, the issues of data quality and confidence have become extremely important. In this environment, errors are promulgated, dispersed, and become difficult to discover and repair. As the number of homonyms and synonyms increases, it becomes crucial to have accurate data identifying various entities. An important aspect of this problem is multilingualism. Multilingual resources such as DBPedia, VIAF, WorldCat,

* This work has been supported by the RFBR grant 14-07-00386.

etc., become increasingly common. When integrating data in different languages, it is necessary to solve the *cross-language identity resolution problem*, in which one data base is in English, and the second one is in Russian. Our experiments have shown that conventional methods of identity resolution usually fail in the cross-language environment. Several named entities with distinct English spellings and translations of their names may correspond in reality to the same Russian entity; on the other hand, several distinct entities may be homonyms and share the same name or some forms of this name. The idea of our approach is to use text analysis methods in combination with attribute-based methods for identity resolution. The dataset of the SB RAS Open Archive and digital library SpringerLink (http://link.springer.com) are used as test examples.

2 Language-specific aspects of identity resolution

One of the projects carried out at the A.P. Ershov Institute of Informatics Systems of the Siberian Branch of the Russian Academy of Sciences (IIS SB RAS) is aimed at populating the Open Archive of the Siberian Branch of the Russian Academy of Sciences (SB RAS Open Archive, Open Archive) [3] with the data of the Open Linked Data cloud (LOD) [4].

The problem is complicated by the fact that the Open Archive uses names written in Cyrillic, and other data sets use Latin names to identify the same persons. Our recent experiments [5] have shown that this problem has *language specific aspects* because sometimes a name in one language can be obtained by translation or transliteration of the name in another language.

Experiments with the RKBExplorer datasets (www.rkbexplorer.com) have shown that a person of the Open Archive has several matches in the RKBExplorer with different spellings and these persons have disjoint lists of their publications. For example, the publications authored or edited by Academician Andrei Petrovich Ershov have been

attributed to 18 distinct persons whose names are Andrei P. Ershov, A.P. Yersh'ov, A. Ershov, and A. Yershov in DBLP RKBExplorer. By checking the DBLP Computer Science Bibliography, the counterpart of RKB Explorer DBLP, three distinct persons with publications belonging to Academician Andrei Petrovich Ershov have been identified. Their names are various forms of the Latin transliteration of "Андрей Петрович Ершов".

Another example is VIAF, the Virtual International Authority File. Its web interface is available on http://viaf.org. The source files of VIAF include some of the most carefully curated files of the names available. In addition, the bibliographic records using the files are professionally created, often reviewed and corrected by many libraries. In spite of the substantial work put into the creation and maintenance of the files, they still have inaccuracies. For example, VIAF has attributed several papers edited by or written by Academician A.P. Ershov to a person identified as http://viaf.org/viaf/196995053 and named Ershov, Aleksandr Petrovich. On the other hand, among the publications attributed to Academician Andrei Petrovich Ershov there are two books on economics (http://viaf.org/viaf/5347110), which can hardly belong to him.

Experiments with WorldCat.org have shown that this resource, too, is not free from identification errors when Russian authors are considered. For example, the list of WorldCat Identities,
(http://www.worldcat.org/wcidentities/lccn-n80162678),
containing descriptions of particularly prominent personalities, has a record dedicated to Academician Andrei Petrovich Ershov. It contains information about the books and papers authored or edited by Academician A.P. Ershov mixed with the publications authored by another A.P. Ershov (Alexander Petrovich) from Novosibirsk. Besides, the articles authored by Alexander Petrovich Ershov and published between 1989 and 2012 have been described as "publications by Andrei Petrovich Ershov published posthumously" (Academician A.P. Ershov died in 1988).

A well-known system for attribute-based identity resolution in the context of the Open Linked Data is SILK [7]. The heuristics used

by VIAF and DBLP for ambiguity detection are described in [8, 9]. Cross-language entity disambiguation requires taking into account differences in orthography between languages and differences in the way these words in different languages are used to express similar meanings. The focus of this paper is on matching the named entities whose English names have been created by a transliteration or translation of Russian names.

Besides, now a large number of publications are digitized, and a most important attribute characterizing each researcher is her or his publications. Nowadays, quite advanced methods of authorship attribution exist, including analysis of character, lexical, syntactic, semantic and application-dependent features [10-12]. The use of these methods is governed by the idea that authors unconsciously tend to use similar lexical, syntactic or semantic patterns. However, when comparing the English texts published by Russian authors, character, lexical, and syntactic methods do not seem to be the most appropriate because different texts of the same author can be translated by different translators who vary in their translation styles. On the other hand, semantic analysis of the texts can reveal, for example, their terminological similarity.

Thus, we suggest a combined use of articles metadata comparison and their text similarity estimation for the cross-language identity resolution. As a source of detailed meta-data, the digital library SpringerLink has been chosen along with full-texts of articles. SpringerLink is currently one of the largest digital libraries that holds more than 9 000 000 documents in various fields of research: computer science and mathematics, life sciences and materials, philosophy and psychology. Its wide range of fields corresponds well to the multidisciplinary orientation of the SB RAS Archive. Besides, SpringerLink contains the full texts of many articles. If the full text of a publication is not available, SpringerLink provides detailed meta-data about the publication, such as ISSN, abstract, the affiliations of its authors (if specified in the text), references etc. Finally, SpringerLink is one of the sources used by a part of the Open Linked Data cloud WorldCat.org (http://worldcat.org).

The content of the SB RAS Open Archive provides various documents (photo documents mainly) reflecting information about people,

research organizations and major events that have taken place in the SB RAS since 1957. The Open Archive contains information about the employments, research achievements, state awards, titles, participation in conferences, academic and social events for each person mentioned in the Archive. The Open Archive has facts about 10 917 persons and 1519 organizations and events. The data sets of the Open Archive are available as an RDF triple store, as well as a Virtuoso endpoint for the SB RAS Archive. Its RDF triple store comprises about 600 000 RDF triples. An example of data of the OpenArchive looks as follows:

<person rdf:about="piu_200809051791">
<name xml:lang="ru">Ершов Андрей Петрович</name>
<name xml:lang="en">Yershov, Andrei Petrovich</name>
<from-date>1931-04-19</from-date>
<to-date>1988-12-08</to-date>
<sex>m</sex>
</person>
<participation rdf:about="w20070417_7_10747">
<participant rdf:resource="piu_200809051791" />
<role-classification>basic</role-classification>
<in-org rdf:resource="fog_pavlovskaya200812254335" />
<from-date>1976</from-date>
<to-date>1988</to-date>
<role xml:lang="ru">заместитель заведующего кафедрой вычислительной математики ММФ</role>
</participation>
<org-sys rdf:about="fog_pavlovskaya200812254335">
<name xml:lang="ru">Новосибирский государственный университет</name>
<org-classification>organization</org-classification>
<from-date>1958-01-09</from-date>
</org-sys>

In the SB RAS Open Archive, all persons are specified by means of a normalized name. The format of a normalized name is <Last-Name, First Name Middle Name>. This attribute has two options: the

Russian-language version and the English-language version. The English version is a transliteration of the Russian version. However, every Russian name can be transliterated in many ways. For example, the Russian family name Ершов can be spelt as Ershov, Yershov, Jerszow, and the first name Андрей can be written as Andrei, Andrey, Andrew.

3 The algorithm for identity resolution

The general scheme of our algorithm is as follows:

1. Our program takes as input a string *R_string,* corresponding to a Russian name from the Open Archive and returns a set of all possible English transliteration and forms variations *E_strings*. *E_strings* can be generated either by Google translate or by our transliteration program.

2. Each generated string $s \in$*E_strings* is used for key word search in SpringerLink. This search results in a list of documents where the key word can occur in the title of article, in the name of organization, in the reference list, etc. All the articles are filtered, and only publications having one of the key words as the author are retained. Each article is specified by a unique identifier. SpringerLink indexes several kinds of data formats (txt, PDF, PNG). For our experiment, however, we convert non-text formats into text and make use of plain text files. A set of metadata such as citation_publisher, citation_title, citation_author, citation_author_institution, etc. are extracted and concatenated to create a text for analysis.

   The authors of the list of articles can be both homonyms and synonyms. We have to process the list of articles and determine which of their authors are synonyms and which of them are homonyms. In other words, the list of articles should be separated into subsets S_1, S_2, S_n such that each subset of articles is authored by a single person and all his or her name variations are synonyms.

3. The publication date and authors' affiliation, provided by SpringerLink are compared with the person's list of affiliations specified by the Open Archive. Again, English names of organizations should be compared against their Russian counterpart. At this stage using transliteration is inappropriate, therefore Google translate and viaf.org web –services are used for generating the English variants of the Russian names of organizations. For example, "Институт Систем Информатики" is identified in VIAF as VIAF ID: 159616561 (Corporate). Permalink: http://viaf.org/viaf/159616561. Its English name provided by VIAF is "A.P. Ershov Institute of Informatics Systems". If there is no English counterpart for the Russian name of an organization, its Google translation is used.

4. To distinguish persons whose affiliations specified in SpringerLink coincide with these indicated in the Open Archive, we generate a matrix that measures pair wise similarity between stemmed affiliations (names of organizations). Cyclic Jaro-Winkler distance [13] is used as the measurement. Based on this comparison, the whole list of articles S is subdivided into three subsets P_1, P_2, and P_3, where P_1 is a set of articles whose affiliations are similar to one of the list of affiliations specified for the given person in the Open Archive, P_2 is a set of articles whose affiliations are not similar to any of the list of affiliations specified by the Open Archive for the given person, and P_3 is a set of articles that have no specified affiliation for the considered author.

5. The program tries to distribute articles without specified affiliations (P_3) among the groups of publications with specified affiliations (P_2). This procedure is based on the text similarity of articles. One of the most effective methods for the semantic analysis of textual data is Latent Dirichlet Allocation with Cullback-Leibler divergency [14]. A simpler and computationally cheaper alternative is to calculate document similarity using the tf-idf weighting with cosine similarity. Before computing the text similarity, the text is cleaned by removing stop words and leaving

only plain content-carrying words, then a stemming procedure [15] is applied. If the similarity value for an article A is below threshold for every group of articles Group 1, ..., Group N, the program creates a new group NewgroupN+1, where N+1 is the serial number of the newly created group.

6. Some articles of subsets P_1 and P_2 are specified by general affiliation such as "Russian Academy of Science". The program tries to distribute articles with more general affiliations among the groups of articles with more specific affiliations. For example, "Siberian Division of Russian Academy of Science" is considered to be more general with respect to "A.P. Ershov Institute of Informatics Systems Siberian Division of Russian Academy of Science". More general affiliation is a substring of a more specific one. If the author's affiliation specified in SpringerLink is considered to be a general name of an organization, the program tries to decide which of the more specific names of the organization can be used as author's affiliation. Text similarity of the articles from groups with the exact names of organizations is used at this stage.

7. Text similarity measure is applied to compare the generated groups of articles and if the similarity value for two groups of articles g_1 and g_2 exceeds the threshold value, the two groups are merged into one.

8. The collection of documents is treated as a graph. Each document is a node identified by its number in the list of documents and every pair of documents is connected by an edge whose weight (W) is given by the similarity between the two documents. A threshold is applied to the similarity matrix to ignore the links between documents with low similarity. The threshold depends on the number of nodes. For example, the threshold is equal to 0.05 for 30 nodes. The obtained graph is drawn by a usual force-directed placement algorithm so that similar documents are placed close to each other. In our case, the force of attraction and the repulsion force both depend on the weight of the edge

between vertices.

Force of attraction= Temperature * SpringForce(d) * W * SpringForceK;

Force of repulsion = Temperature * ElectricForce(d) / W * ElectricForceK;

SpringForce(d) = 2 * log(d),

ElectricForce(d) = 1 / d^2, where d is the distance, and W is the similarity between two vertices.

The result of the program is a set of SpringerLink articles subdivided into several groups. The first group of articles is attributed to the person described in the Open Archive.

4 Results of experiments

Experiments have shown that name variations generated by our program were more appropriate than that of Google translate. For example, 408 English variants have been generated by our program for the Russian name "Валерий Александрович Непомнящий," among which only five variations have been discovered in Springerlink: V.A. Nepomnyashchii, Valery Nepomniaschy, V.A. Nepomniaschy, Valery A. Nepomniaschy, V.A. Nepomnyaschy. Google Translate has created 160 variants of the same name but some name variations existing in SpringerLink were absent from Google translate results. For example, the name variation "V. A. Nepomnyaschy" existing in SpringerLink was generated by our program but was not generated by Google Translate.

A weak point of Google Translate was that along with the transliteration of names, it generated their translations. For example, Google translate generated variants like "Valery oblivious" for the Russian name "Валерий Непомнящий", and "Vadim cats" for the Russian name "Вадим Котов".

An example of the program, searching for the articles authored by the Russian person "В.А. Непомнящий" in the digital library SpringerLink is shown in Fig.1. English variations of a Russian personal name

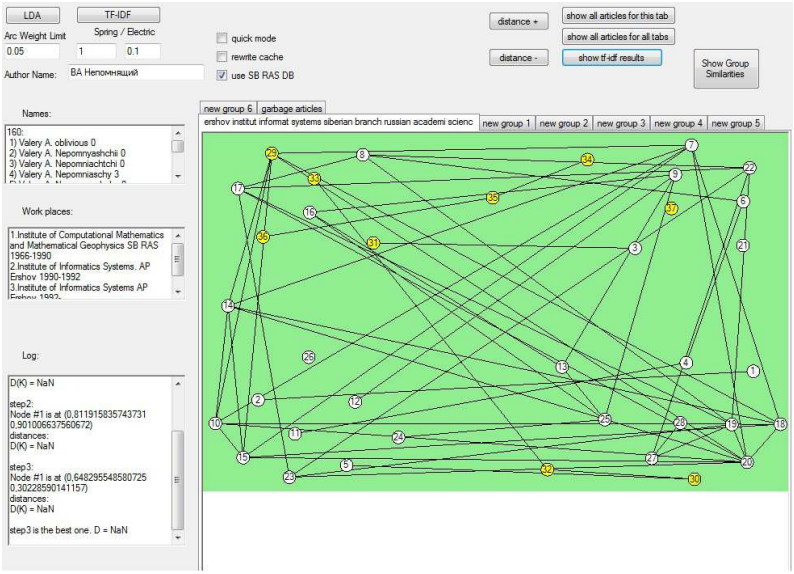


Figure 1. A placement of several articles attributed to the person named as Непомнящий, Валерий Александрович in the Open Archive. Articles with specified affiliation are shown in a lighter color

are displayed in the upper left tab. English versions of affiliations for the given person are displayed in the middle left tab. In the center, a graph representing the publications attributed by the algorithm to the person from the Open Archive is shown. Each node of the graph corresponds to an article of the SpringerLink digital library. Light nodes correspond to the articles with a specified affiliation and dark nodes represent articles without a specified affiliation. The total of 45 publications have been found for the "BA Непомнящий" query. There were 15 papers belonging to a person named as V. A. Nepomnyashchii, 3 papers belonging to a person named as Valery Nepomniaschy, 23 papers belonging to a person named V. A. Nepomniaschy, 3 papers belonging to a person named as Valery A. Nepomniaschy, and 1 paper belonging

to V. A. Nepomnyaschy. All these papers were written by two real persons. The first person (described in the Open Archive) has used all the above variants of his name including V. A. Nepomnyashchii, and the second person has always named himself as V. A. Nepomnyashchii. The program has correctly identified 37 publications authored by Valery Nepomniaschy from the Open Archive but the publications of two of his homonyms were placed in distinct groups.

5 Conclusion

The program has been tested on a test sample of 100 persons employed by the IIS SB RAS (about 3,000 publications) and on the articles authored by Academician A.P. Ershov. To verify this approach, we have compared the data extracted from the SpringerLink digital library with the data of the Academician A. Ershov's archive and the digital library eLIBRARY.RU. Regarding the SpringerLink articles, a significant variation in the amount of available texts is detected (from a few lines to a few dozens of pages), which significantly affects the accuracy of identification. About eighty percent of the analyzed articles in SpringerLink had no information on the full names of their authors (only short forms were given) and approximately seventy percent of author's affiliations have been provided. Nevertheless, a joint comparison of attributes and text similarities have shown good accuracy, close to 93 percent.

References

[1] Anderson A. Ferreira, Marcos André Gonçalves, Alberto H. F. *Laender Disambiguating Author Names in Large Bibliographic Repositories.* In: International Conference on Digital Libraries, New Delhi, India, 2013.

[2] Yang Song, Jian Huang, Isaac G. Councill, Jia Li C. Lee Giles. *Efficient Topic-based Unsupervised Name Disambiguation.* Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries, 2007, pp. 342–351.

[3] A.G. Marchuk, P.A. Marchuk. *Specific features of digital libraries construction with linked content.* Proc. of the RCDL'2010 Conf. – 2010. pp. 19–23. (In Russian).

[4] A. Schultz et al. *How to integrate LINKED DATA into your application* //Semantic technology & Business Conference, San Francisco, June 5, 2012. http://mes-semantics.com/wp-content/uploads/2012/09/Becker-etal-LDIF SemTechSanFrancisco.pdf.

[5] Z.V. Apanovich, A.G. Marchuk. *Experiments on using the LOD cloud datasets to enrich the content of a scientific knowledge base.* P.Klinov and D.Mouromtsev (Eds.) KESW 2013, CCIS 394, Springer Verlag Berlin Heidelberg 2013, pp. 1–14.

[6] C. J. Godby, R. Denenberg. *Common Ground: Exploring Compatibilities Between the Linked Data Models of the Library of Congress and OCLC.* http://www.oclc.org/research/publications/2015/oclcresearch-loc-linked-data-2015.html.

[7] R. Isele, A. Jentzsch, Ch. Bizer. *Silk Server – Adding missing Links while consuming Linked Data.* 1st International Workshop on Consuming Linked Data (COLD 2010), Shanghai, November 2010.

[8] M. Ley. DBLP - Some Lessons Learned. PVLDB 2(2), 2009, pp. 1493–1500.

[9] T. B. Hickey, J. A. Toves. *Managing Ambiguity In VIAF.* D-Lib Magazine 20 (July/August), 2014. doi:10.1045/july2014-hickey.http://www.dlib.org/dlib/july14/hickey/07hickey.html.

[10] E. Stamatatos. *A survey of modern authorship attribution methods. Journal of the American Society for Information Science and Technology.* 60(3), pp. 538–556, 2009.

[11] A.A. Rogov, Yu. Vl. Sidorov. *Statistical and Information-calculating Support of the Authorship Attribution of the Literary Works.* Computer Data Analysis and Modeling: Robustness and Computer Intensive Methods: Proc. of the Sixth International

Conference (September 10-14, 2001, Minsk). Vol.2: K-S/ Edited by Prof. Dr. S. Aivazian, Prof. Dr. Yu. Kharin and Prof. Dr. H. Rieder. Minsk: BSU, 2001, pp. 187–192.

[12] O. Kukushkina, A. Polikarpov, D. Khmelev. *Using literal and grammatical statistics for authorship attribution.* Probl. of Info. Trans. 37(2), pp. 172–184, 2001.

[13] W. W. Cohen, P. D. Ravikumar, S. E. Fienberg. *A Comparison of String Distance Metrics for Name-Matching Tasks.* IIWeb 2003, pp. 73–78.

[14] D. M. Blei, A. Ng, M. Jordan. *Latent Dirichlet allocation Journal of Machine Learning Research.* (3) 2003, pp. 993–1022.

[15] M. Steyvers, T. Griffiths. *Probabilistic Topic Models Handbook of Latent Semantic Analysis.* 2007.

[16] http://www.codeproject.com/Articles/12098/Term-frequency-Inverse-document-frequency-implemen

[17] http://snowball.tartarus.org/

Zinaida Apanovich, Alexander Marchuk, Received July 9, 2015

Zinaida Apanovich
1) A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences
6, Acad. Lavrentjev pr., Novosibirsk 630090, Russia
Phone: +7 383 3308652
E–mail: apanovich@iis.nsk.su
2) Novosibirsk State University
630090, Novosibirsk-90, 2 Pirogova Str.

Alexander Marchuk
1) A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences
6, Acad. Lavrentjev pr., Novosibirsk 630090, Russia
Phone: +7 383 3308652
E–mail: mag@iis.nsk.su
2) Novosibirsk State University
630090, Novosibirsk-90, 2 Pirogova Str.

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Contrastive diachronic study on Romanian language

Daniela Gîfu

Abstract

The paper describes a study based on diachronic exploration of Romanian texts in order to implement a technology for detecting automatically the morpho-lexical features from 1840 to nowadays. The aim of this survey is to analyse the morphology and lexical-semantics of Romanian language, based on a few Romanian and Moldavian resources, and an algorithm to explain how the language evolved. We investigated the use of a lexicon-based and a learning-based approach to identify morpho-syntactic and semantic items to reconstruct the general trends in the Romanian language evolution. The study is meant to help linguists, specialists and non-specialists in NLP.

Keywords: diachronic, lexicon, morphology, lexical-semantic, language model.

1 Introduction

This survey is based on diachronic exploration of Romanian texts in order to implement a technology for detecting automatically the morpho-lexical features from 1840 to nowadays. The aim of this work is to investigate the linguistic crisis that affects the multilingual Republic of Moldova in parallel with the Romanian language, using natural language processing (NLP) methodology for tracking diachronic changes in the 19th century.

In this case, the methodology relied on part-of-speech (POS) tagging, often with manual post-editing. Most previous works (e.g. Leech

et al., 2009; Davies, 2013) has focused mainly on the linguistic interpretation of the results.

We cannot imagine the human thinking without a system of signs. Thinking is dependent on the meanings and concepts that are organized as a sign interpreter. The linguistic sign is an unmotivated conventional connection between a signifier (sound or graphic body of a word) and a signified concept (i.e. mental representation of an object) (Saussure, F. de, 1971: p. 99), a concept (Morris, 1971: 19), that is not a real object (Vasiliu, 1995: 13).

Ch. S. Peirce (Peirce, 1990: 133) considers that the sign has a ternary structure. He established a relationship between a real object or concept and an interpreter, which is the effect that the sign has on the receiver, representing the image of the object induced in his / her consciousness. In fact, according to Peirce, the sign will have an infinite number of interpreters that approximate more accurately the meaning of the sign.

Our research is based on the premise that *the semantics refers to those types of signs, which are artificially or naturally organized in language.*

This paper is structured as follows: section 2 presents some approaches for lexical semantics. Section 3 proposes a specific methodology for research of language using language technology, and section 4, the conclusions section emphasizes the importance of computational tools for analysis of contrastive diachronic studies on the Romanian language and directions for future work.

2 Background

Since mutual replacement in a context is a criterion for perfect synonymy, a recent study develops an automated method of aligning paraphrases contexts to detect synonymy (Grigonyt et al., 2010). Scientists show preoccupations to add derivative links to the WordNet and their comparison in more languages (Gala et Mititelu, 2013). Iftene and Balahur (Iftene and Balahur, 2007) used WordNet (Fellbaum, 1998) to look up synonyms for a word and try to map them to nodes from the

text tree.

Another important approach for lexical semantics is nocuous ambiguity which occurs when a linguistic expression is interpreted in different ways by readers in a context. A study made by Hui Yang (Yang, H. et al. 2010) presents a method to automatically identify ambiguity or harmful substance, which is likely to lead to misunderstandings. The model is built on a learning machine architecture which learns from a set of heuristics. These heuristics predict a factor that can make the reader to favor one interpretation.

Regarding the current trends in assessing the performance of the systems to eliminate the ambiguity in the meanings of words oriented to application, Lefever et Hoste built a multilingual reference data set for a multilingual disambiguation of meanings. The data set was created for a sample of 25 lexical nouns in English, for which translations were recovered in 5 languages, especially in Dutch, German, French, Italian and Spanish. (Lefever et Hoste, 2010). Tufiş, also proposed a method for the automatic recognition of the different meanings of polysemantic words (Tufiş, 2002).

Numerous studies relate to implementation of semantic annotations of various types. Garcia and colleagues (Garcia et al., 2012) present the methodology and the results of the analysis of terms related to verbs or nouns processing in a corpus of specialized texts dealing with ceramics. This method is useful in distinguishing the different meanings of the same verb.

In a recent article (Eckle-Kohler et al. 2012) solutions are sought for compatibility of different formats of resources. UBY-LMF LMF is a model based on the large-scale lexical-semantic heterogeneous resources in multiple languages. UBY-LMF allows standardization LSRS to a level of fine grain lexical information by using a large number of data and the categories from ISOC. UBY-LMF was evaluated by transforming LSRS in two languages of WordNet format in English. Heterogeneous resources are: Wiktionary, Wikipedia, OmegaWiki, FrameNet, VerbNet. German Wikipedia, GermaNet.

Preocupations in the direction of reconstructing a diachronic morphology for Romania already exist (Cristea et al., 2012). They are

based on the digital version of the Romanian Language Thesaurus Dictionary (eDTLR) (Cristea et al, 2007), the most important lexical resource. The authors detected the old form words occurring in the citations. This report presents a diachronic text classification, another important issue in NLP literature, (Mihalcea and Năstase, 2012; Popescu and Strapparava, 2015) in order to evaluate approaches toward diachronic text analysis of a two corpora of Romanian language from Romania and Republic of Moldova.

3 Work Methodology

In this section we describe the sequence of operations for processing texts of the Romanian language (published in Romania and in the Republic of Moldova) and we show how methods for automatic classification of unstructured data, the WEKA classifiers, can be used.

3.1 Corpus

The corpus used to support the work described in this report consists of two novels, published in Romania and Moldova before 1900s: "Alexandru Lăpuşneanu" (1840), by Costantin Negruzzi, and "Polidor and Hariti" (1843), by the equerry Dimitrie Balica. Note that the text was written in Cyrillic, then transcribed in Latin (Marşalcovschi et al., 2012), but the transliteration from Cyrillic in Latin is not on interest in this paper.

3.2 Pre-processing

The Romanian automatic pre-processing chain applied on raw texts of the book consists of the following tasks, executed in sequence: segmentation, tokenization, lemmatization, and part-of-speech (POS) tagging, described below.

First, it was necessary to extract both texts from PDF[1], and we removed the footnotes and the symbol "u", which has only a phonetic

[1] http://online2pdf.com – free services

role).

Ex. 1: ianu patru partu => ian patr part

Ex.2: dacoromanomoldavu => dacoromanomoldav

Ex.3: oarua => oara

(A) Segmentation (splitting the text in sentences).

(B) Tokenization (demarcates words or word components, but also numbers, punctuation marks, abbreviations, etc.).

(C) Lemmatization (determines lemmas of words).

(D) Part-of-speech tagging (identifies POS categories and morpho-syntactic information of tokens).

Note that the tokenization, lemmatization and POS tagging are realized in one step, using http://nlptools.infoiasi.ro/WebPosRo/.

(E) To obtain the XML format of each text, the free service http://wsdlbrowser.com was used. Because, the web service has problems with the symbols ,,<" and ,,>", they have also been removed from the text.

(F) To remove the unknown words from each XML, we applied a few regular expressions to get a list of unfamiliar words, separated by commas:

Extracting of words that are labeled ,,NotInDict" from XML files:

a. Replacing .*NotInDict.*?>(.*)<.* with $1

b. Replacing <.* with ,,"

Removing of extra spaces: replacing "\s+"" with " ", and duplicate words in Javascript.

For example, here is a XML fragment for each novel:

(1) "Polidor and Hariti" contains 26972 words and 2771 unknown words.

For instance: *mai ianti acistu al mieu...* <?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<POS_Output>

<S> ...

<W LEMMA="mai" MSD="Rg" POS="ADVERB" id="null.242" offset="1323">mai</W>

<W Case="direct" Definiteness="no" EXTRA="NotInDict" Gender="masculine" LEMMA="ianti" MSD="Afpmprn" Number="plural" POS="ADJECTIVE" id="null.243" offset="1327">ianti</W>

<W Case="direct" Definiteness="no" EXTRA="NotInDict" Gender="masculine" LEMMA="acistu" MSD="Afpmsrn" Number="singular" POS="ADJECTIVE" id="null.244" offset="1334">acistu</W>

<W Case="direct" Gender="masculine" LEMMA="al" MSD="Tsmsr" Number="singular" POS="ARTICLE" Type="possessive" id="null.245" offset="1341">al</W>

<W EXTRA="NotInDict" LEMMA="mieu" MSD="Y" POS="ABBREVIATION" id="null.246" offset="1344">mieu</W>

...

< /S>

< /POS_Output>

(2) "Alexandru Lăpuşneanu" contains 7894 words and 173 unknown words.

For instance: *O gvardie numeroasă de lefecii albanezi, ...*

<W Case="direct" Gender="feminine" LEMMA="un" MSD="Tifsr" Number="singular" POS="ARTICLE" Type="indefinite" id="null.1613" offset="7713">O</W>

<W Case="direct" Definiteness="no" EXTRA="NotInDict" Gender="feminine" LEMMA="gvardie" MSD="Ncfsrn" Number="singular" POS="NOUN" Type="common" id="null.1614" offset="7715">gvardie</W>

```
<W Case="direct" Definiteness="no" Gender="feminine"
LEMMA="numeros" MSD="Afpfsrn" Number="singular"
POS="ADJECTIVE" id="null.1615"
offset="7723">numeroas</W>

<W LEMMA="de" MSD="Sp" POS="ADPOSITION"
id="null.1616" offset="7733">de</W>

<W Case="direct" Definiteness="no" EXTRA="NotInDict"
Gender="feminine" LEMMA="lefecii" MSD="Ncfprn"
Number="plural" POS="NOUN" Type="common"
id="null.1617" offset="7736">lefecii</W>

<W Case="direct" Definiteness="no" Gender="masculine"
LEMMA="albanez" MSD="Afpmprn" Number="plural"
POS="ADJECTIVE" id="null.1618"
offset="7744">albanezi</W>

...

< /S>

< /POS_Output>
```

(G) Extracting the proper nouns from each XML in order to improve NER (*Name Entity Recognition*) with old person names / toponims. Actually, since 2014, NER is improving with the Geonames lists (Sălăvăstru and Gîfu, 2015).

Here is one example for each novel, where it is shown the new attribute OLD_MDV="TRUE" for each old toponim/person name extracted from texts of the Republic of Moldova:

(1) "Polidor and Hariti" contains 257 nouns, which have the first capital letter. After the manual verification, the list of proper nouns contains 89 (a few proper nouns appear inflected).

For instance:

```
<ENTITY TYPE="LOCATION" SUBTYPE="TOWN"
OLD_MDV="TRUE"/> Bălţălor</ENTITY>
```

Note, Bălţi (here, Bălţălor) is a town from the Republic of Moldova.

(2) "Alexandru Lăpuşneanu" contains 24 nouns, which have the first letter capitalised. After the manual verification, the list of proper nouns contains 13 of them (two proper nouns appear inflected). We include a new attribute OLD_RO="TRUE" for each old toponim / person name extracted from texts of Romania.

Some examples:

<ENTITY TYPE="LOCATION" SUBTYPE="COUNTRY" OLD_RO="TRUE"/> Valahia</ENTITY>

4 WEKA Statistics

Because these novels have different dimensions, here we take a sample from "Polidor and Hariti" with an equal number of unknown words (UN) (173, the number which was extracted from "Alexandru Lăpuşneanu"). Below, it is presented this corpus:

Table 1. Unknown words in the testing corpus resulted automatically and manually

"Polidor and Hariti"					**"Alexandru Lăpuşneanu"**				
Total words	Automatically		Manually		Total words	Automatically		Manually	
	Unknown words	%	UN	%		Unknown words	%	UN	%
653	173	26.49%	168	25.73%	7894	173	2.19%	171	2.17

1. Those two lists of unknown words were edited in Notepad ++ and then saved as TXT. The TXT format was changed in CSV format, file type supported by WEKA.

Here is a sample file which contains unknown words arranged by period of time and the place.

```
Period, Words
< 1900_RO,vreu
< 1900_RO,perdut
...
< 1900_Bessarabia, întaia
< 1900_Bessarabia, gravure
```

2. CSV file will be processed by WEKA (*Waikato Environment for Knowledge Analysis*).

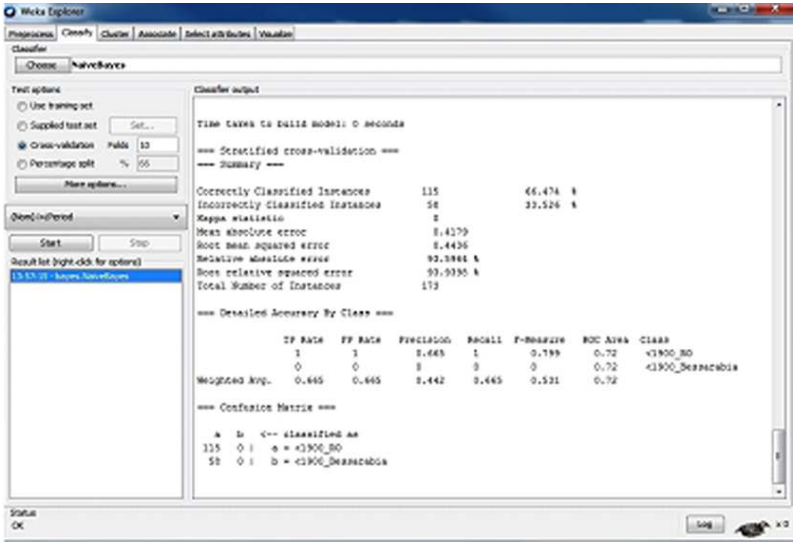


Figure 1. Work session – Naïve Bayes classifier in WEKA

3. In this paper a type of classifications is presented.

a. before 1900, "Alexandru Lăpuşneanu" for Romania and a sample from "Polidor and Hariti" for Bessarabia.

b. From the XML corpora, the attributes LEMMA, POS and MSD were extracted using a Perl script.

4. In this report, it was preferred the Naïve Bayes classifier (see Fig. 1).

   Above is a sample output (173 instances) for a Naïve Bayes classifier, using 10-fold cross-validation. It could be explained as follows:

   So the percentages and raw numbers add up,

   ```
   aa + bb = 115 + 0 = 115
   ```

   ```
   ab + ba = 58 + 0 = 58
   ```

   The percentage of correctly classified instances is often called accuracy or sample accuracy.

   ```
   correctly classified instances = 66,47%
   ```

   Kappa is a chance-corrected measure of agreement between the classifications and the true classes.

   ```
   kappa = 0
   ```

   Note that a classifier is doing better than chance if the kappa value is greater than 0.

   The error rates are used for numeric prediction rather than classification. In numeric prediction, predictions aren't just right or wrong, the error has a magnitude, and these measures reflect that.

   ```
   Mean absolute error = 0.41
   ```

   ```
   Root mean squared error = 0.44
   ```

   ```
   Relative absolute error = 93.59%
   ```

   ```
   Root relative squared error = 93.93%
   ```

(1) For <1900_RO

`TP Rate = 1`, being the rate of true positives (instances correctly classified as a <1900_RO class).

`FP Rate = 1`, being the rate of false positives (instances falsely classified as a <1900_RO class).

`P = 0.66`, being the proportion of instances that are truly of a <1900_RO class divided by the total instances classified as that class.

`R = 1`, being the proportion of instances classified as a <1900_RO class divided by the actual total in that class (equivalent to TP rate).

`F-Measure = 2 * Precision * Recall / (Precision + Recall) = 0.79`, being the harmonic average

(2) For <1900_Bessarabia

`TP Rate = 0`

`FP Rate = 0`

`P = 0`

`R = 0`

`F-Measure = 0`

In both cases, `ROC = 0.72`

Note that ROC indicator is considered being one of the most important values output by Weka. An optimal classifier will have ROC area values approaching 1, with 0.5 being comparable to "random guessing" (similar to a Kappa statistic of 0).

No.	Label	Count
1	< 1900_RO	57
2	< 1900_Bessarabia	58
Total instances		115

5. The lines from CSV file were mixed. Before that, those two lists resulted automatically were manually corrected: 2 attributes (Period & Words).

   The Naïve Bayes results are:

   Above is a sample output for a Naïve Bayes classifier, using 10-fold cross-validation. It could be explained as follows:

   Here there were 115 instances.

   So the percentages and raw numbers add up,

   ```
   correctly classified instances = 47,82%
   kappa = -0.04
   Mean absolute error = 0.50
   Root mean squared error = 0.50
   Relative absolute error = 99.98%
   Root relative squared error = 99.98%
   ```

   (1) For <1900_RO

   ```
   TP Rate = 0.26
   FP Rate = 10.31
   P = 0.45
   R = 0.26
   F-Measure = 0.33
   ```

   (2) For <1900_Bessarabia

   ```
   TP Rate = 0.69
   FP Rate = 0 .73
   P = 0.48
   R = 0.69
   F-Measure = 0.57
   ```

   In both cases, ROC = 0.46

5 Conclusions and discussions

So far, the proportion of unknown words, for the period before 1900, reflects that the influence of other languages (e.g. Russian) on the Moldavian language is more obvious (a strong connection between linguistic and national identities) than of Romania. The vocabulary, the phoneme and the structure of Balica's work did not meet the literary norms already existent in Romania. In Bessarabia occupied by Russians in 1812 and then annexed to the Tsarist Empire, the educated literature in Romanian was missing. Only Culrescu tempted to write a story about outlaws (Pushkin, who was here in exile, praised this subject). The population from here practiced a typological rural style of Romanian language and folk literary traditions.

References

[1] Cristea, D., Simionescu, D., Haja, G. (2012). Reconstructing the Diachronic Morphology of Romanian from Dictionary Citations. In Proceedings of LREC-2012, Istanbul, 21-25 May.

[2] Cristea, D., Răschip, M., Forăscu, C., Haja, G., Florescu, C., Aldea, B., Dănilă, E. (2007). *The Digital Form of the Thesaurus Dictionary of the Romanian Language*. In Proceedings of SpeD 2007 Speech Technology and Human - Computer Dialogue, Iaşi, May 10-12, 2007.

[3] Davies, M. (2013). *Recent shifts with three non-finite verbal complements in English: Data from the 100-million-word Time corpus (1920s-2000s)*. In: Aarts, Close, Leech and Wallis (eds.) The verb phrase in English: Investigating recent linguistic change with corpora, Cambridge: Cambridge University Press. pp. 46–67.

[4] Eckle-Kohler, J., Gurevych, I., Hartmann, S., Matuschek, M. et Meyer, C. M. (2012). *UBY-LMF – A Uniform Model for Standardizing Heterogeneous Lexical-Semantic Resources in ISO-LMF* in Proceedings of LREC 2012, p. 275–282.

[5] Fellbaum, C. et al. (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

[6] Gala, N. et Barbu-Mititelu, V. (2013). *Lex-Rom: un réseau lexical pour les familles morphologiques dans les langues romanes,* Congrès international de Linguistique et Philologie Romanes, Nancy.

[7] García, N. M., L'Homme, M. C., Alcina, A. (2012). Semantic Relations Established by Specialized Processes Expressed by Nouns and Verbs: Identification in a Corpus by means of syntactico-semantic Annotation in Proceedings of LREC 2012, pp. 3814–3819.

[8] Grigonyt, G., Cordeiro, J., Dias, G. et Moraliyski, R., (2010), *Paraphrase Alignment for Synonym Evidence Discovery,* in Colling, Beijing, China vol. 2, pp. 403–410.

[9] Iftene, A. And Balahur-Dobrescu, A. (2007). Hypothesis Transformation and Semantic Variability Rules Used in Recognizing Textual Entailment. In Proceedings of the Workshop on Textual Entailment and Paraphrasing, Prague, pp. 125–130.

[10] Leech, G., Hundt, M., Mair, C. and Smith, N. (2009). *Change in Contemporary English: A Grammatical Study*. Cambridge: Cambridge University Press.

[11] Lefever, E., Hoste, V. (2010). Construction of a Benchmark Data Set for Cross-lingualWord Sense Disambiguation in Proceedings of LREC, 2010, pp. 1584–1590.

[12] Marşalcovschi, T.-T., Abramciuc, M., and Harconiţa, E. (2012). *Manuscrise* by the equerry Dimitrie Balica, Bălţi.

[13] Mihalcea, R. and Nstase, V. (2012). *Word epoch disambiguation: Finding how words change over time*. In Proceedings of ACL 2012.

[14] Morris, Ch. (1971). *Writings on the General Theory of Signs*, The Hague, Paris, Mouton.

[15] Peirce, Ch. S. (1990). *Semnificaţie şi acţiune*, Bucureşti, Ed. Humanitas.

[16] Popescu, O. and Strapparava, C. (2015). Semeval-2015 task 7: *Diachronic text evaluation*. In Proceedings of SemEval 2015.

[17] Saussure, F. de (1971). *Cours de linguistique générale*, Ediţia a III-a, Paris, Payot.

[18] Sălăvăstru, A. and Gîfu, D. (2015). Annotating Geographical Entities at the *16th International Conference on Intelligent Text Processing and Computational Linguistics* (CICLing 2015), 14-20 Apr. 2015, Cairo, Egypt.

[19] Tufiş, D. (2002). Dezambiguizarea semantic automat în corpusuri paralele. In Limba Român în Societatea Informaţională – Societatea Cunoaşterii, Ed. Expert, Tufiş, D., Filip, F. Gh. (coord.), pp. 237–270.

[20] Vasiliu, Em. (1995). *Elemente de filosofie a limbajului*, Bucureşti, Editura Academiei.

[21] Yang, H., Roeck, A., Willis, A., Nuseibeh, B. (2010). *A Methodology for Automatic Identification of Nocuous Ambiguity* in Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), vol. 2, pp. 1218–1226.

Daniela Gîfu[1,2], Received July 12, 2015

[1] Faculty of Computer Science, "Alexandru Ioan Cuza" University of Iaşi
[2] Center for Advanced Research in Applied Informatics,
University of Craiova
E–mail: daniela.gifu@info.uaic.ro

Part 9

Cryptography and security

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

About Vigenere cipher modifications

Eugene Kuznetsov

Abstract

The aim of this work is a modification of the classical Vigenere cipher, in order to improve the statistical properties of the cipher-text obtained after operation. The work deals with the (third) modification of the classic Vigenere cipher. Classic code is modified to a state in which it can already be implemented and analyzed. The necessary information from the theory of algebraic systems (fields, near-fields, groups, quasigroups, Latin squares, orthogonal tables etc.) is provided. Using the properties of these algebraic systems the modification of the cipher is constructed and studied.

Keywords: Vigenere cipher, quasigroup, orthogonal tables.

1 Introduction

This work is dedicated to one of the most important aspects of information security software – encryption methods. There are many reliable encryption algorithms now, but most of them have a significant drawback – low speed of work. In this paper the famous Vigenere cipher will be discussed. The mere cipher is not of interest today, because there are simple hacking methods. But the principles laid down in it, potentially allow us to create quick and at the same time robust ciphers.

The aim of this work is a modification of the classical Vigenere cipher, in order to improve the statistical properties of the cipher-text obtained after operation. The basis of modification of the classic cipher is an encryption method by bigrams. Its essence lies in the fact that the original message is divided into pairs and each pair of symbols according

to a certain law (special sequence table or tables) is encrypted in some other pair of symbols.

The work deals with the (third) modification of the classic Vigenere cipher. Classic code is modified to a state in which it can already be implemented and analyzed. The necessary information from the theory of algebraic systems (fields, near-fields, groups, quasigroups, Latin squares, orthogonal tables etc.) is provided below. Using the properties of these algebraic systems the modification of the cipher is constructed and studied. Actually, these tables are a "chip" method, so they are paid a lot of attention.

2 Modified Vigenere cipher.

2.1 Polyalphabetic ciphers, Vigenere cipher.

Vigenere cipher is a multi-alphabet advanced encryption system. The idea of the cipher is to use as the key the text of an unencrypted message or an encrypted text. This cipher Vigenere described in his book "A Treatise of ciphers." In its simplest form the basis of the table was taken Trithemius table which subsequently dubbed as the Vigenere's table.

Vigenere's table consists of the alphabet shifted cyclically to the left by one character, but other permutations are available too. Additionally, the first line may be a randomly mixed alphabet.

The encryption process is as follows: plain text (which must be encrypted) is written in a line with no spaces. Next, you must determine the key. Vigenere proposed to use as a key the plain text itself, adding to the top of the key a random selected symbol. But as a key it is possible to use any other sequence of characters equal in length to the plaintext.

To produce the cipher-text we take the first letter of the plaintext as an index row in a table Vigenere and standing beneath the letter – as a column. At the intersection of the pair of tables write out the character of the cipher-text. Then repeat these steps for each of the remaining characters.

In order to decrypt the plaintext, you must know the cipher-text and the key. Take the first letter of the key, define the corresponding column in the Vigenere's table and run through it from top to bottom, until you meet the first character of the cipher-text. Once the desired character is met, we write a letter indicating this line, so we get the first character of the plaintext. We do the same steps for the remaining characters of the key and the cipher-text.

In practice, in the programming of the encryption algorithm it is not necessary to have the Vigenere's table in memory, since the encryption algorithm can be represented by some algebraic formula based on such specific algebraic structures, as a field, near-field, orthogonal pair etc.

2.2 Algebraic concepts.

Hacking classic Vigenere's cipher strongly relies on the presence of a codeword and its length. Therefore, if we save (slightly modified) an encryption method by bigrams, but to refuse from the code word, then the usual method of hacking will not act.

Definition 1. ***Latin square*** *of order n is a square table $n \times n$, where each row and each column contains numbers from 1 to n, and each number is found exactly once.*

Definition 2. *The system $\langle G, \cdot \rangle$ is called a* ***quasigroup*** *if the following properties hold:*

1. *"$\cdot$" is a binary operation defined on the set G;*
2. *Each of the equations $x \cdot a = b$ and $a \cdot x = b$ has exactly one solution in G for any $a, b \in G$.*

From the algebraic viewpoint Latin square is a "multiplication table" of a quasigroup.

Definition 3. *A table of order n is called a* ***selector*** *if it satisfies one of the following conditions: $x \cdot y = x$ or $x \cdot y = y$. In the first case the selector is called a* ***right*** *selector, in the second case – the* ***left*** *selector.*

If we take an arbitrary Latin square and a selector of corresponding dimension, the resulting pair of tables will have the property of orthogonality. That is, upon imposition of one of them to another, we obtain a table of pairs of symbols in which each pair of symbols appears exactly once. Algebraically this orthogonal property is described by the following definition.

Definition 4. *Two operations $\langle Q, \cdot \rangle$ and $\langle Q, \circ \rangle$ on the same set Q are called **orthogonal** (or forming an **orthogonal pair**) if the following system*

$$\begin{cases} x \cdot y = a, \\ x \circ y = b, \end{cases}$$

has exactly one solution in Q for any $a, b \in Q$.

Definition 5. *A **near-field** is a set Q with two binary operations "$+$" (addition) and "$\cdot$" (multiplication) defined on it, satisfying the following axioms:*

1. *$\langle Q, + \rangle$ is a commutative group;*
2. *$(a \cdot b) \cdot c = a \cdot (b \cdot c)$ for all elements $a, b, c \in Q$;*
3. *$(a + b) \cdot c = a \cdot c + b \cdot c$ for all elements $a, b, c \in Q$;*
4. *The set Q contains an element 1 such that $1 \cdot a = a \cdot 1 = a$ for every element $a \in Q$;*
5. *For each non-zero element $a \in Q$ there exists an element a^{-1} such that $a \cdot a^{-1} = 1 = a^{-1} \cdot a$.*

Definition 6. *If in the near-field Q the multiplication operation "$\cdot$" is commutative ($a \cdot b = b \cdot a$), then the resulting near-field is called a **field**.*

From the history of orthogonal Latin squares the following method of constructing a sufficiently large set of mutually orthogonal squares of order n is known (but only when $n = p^k$, where p is a prime number, and k is a positive integer).

Let $\langle Q, +, \cdot, 0, 1\rangle$ be a near-field of order n. For any $a \in Q$ we define a new operation $x \overset{a}{\cdot} y$ by the formula:

$$x \overset{a}{\cdot} y = a \cdot x + (1 - a) \cdot y.$$

This operation has the following properties:

1. $x \overset{a}{\cdot} y$ is a quasigroup, if $a \neq 0, 1$;
2. Operations $x \overset{a}{\cdot} y$ and $x \overset{b}{\cdot} y$ are orthogonal for any $a \neq b$.

Let the operations "$+$" and "$\cdot$" are set; then for the generation of orthogonal $(n \times n)$-tables we can use the formula

$$x_{ij} = a \cdot i + (1 - a) \cdot j,$$

where $i, j \in \{0, 1, ..., n - 1\}$, and $a \in \{2, ..., n - 1\}$.

2.3 Procedures for encryption and decryption.

The encryption procedure by bigrams is similar to the encryption process of the classical Vigenere's cipher, only the first bigram symbol is taken from the first table and the second bigram symbol is taken from the second table (instead of a key sequence, as it was done in the classic Vigenere's cipher). In other words, if we take the table of pairs resulting in the superposition of two orthogonal tables mentioned above, then the plaintext bigram (x, y) corresponds to the encryption bigram (a, b), which is located at the intersection of the x-th row and y-th column. This procedure is repeated sequentially for all bigrams of the encrypted text.

Latin square in the algorithm described above can be changed to another Latin square. Orthogonality with the selector remains, and the encryption procedure does not change. The sequence of these squares (or its generation by any algebraic method) is defined by the secret key (or by periodic key sequence). It is easy to see that the statistical hacking algorithms stop working when the number of squares becomes substantially greater than 2.

It is easy to notice that the second character of bigram always remains the same after the procedure encryption. This may facilitate the probable hacking of this cipher. To avoid this we must use another Latin square (or $(n \times n)$-table) instead of the selector. It is important only that these two $(n \times n)$-tables will be orthogonal.

To eliminate hack statistical methods it can be used several different tables instead of a single one. Then it is obvious that if more different tables to be used, then statistics of a source text will be violated stronger. The effect will be exactly the same as the increase in the length of a code phrase in the classic Vigenere's cipher.

References

[1] D. Kahn. *The First 3,000 Years // The Codebreakers – The Story of Secret Writing.* — New York: Charles Scribner's Sons, 1967, 473 p.

[2] S. Singh. *The Evolution of Secret Writing // The Code Book – The Secret History of Codes & Code-breaking.* – London: Forth Estate, 2000, pp. 3–14.

[3] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone. *Handbook of Applied Cryptography,* 2002.

[4] E. Kuznetsov, S. Novoseltsev. *A modification of Vijener's cipher by the methods of non-associativity algebra.* – ASADE-2007, Abstracts, Chisinau, August 21-23, 2007, 86.

Eugene Kuznetsov Received July 12, 2015

Eugene Kuznetsov
Institution: Institute of Mathematics and Computer Science, Academy of Sciences, MOLDOVA
Address: MD-2028, Academiei str., 5, Chisinau, MOLDOVA
Phone: (373) 022 738029
E–mail: kuznet1964@mail.ru

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Stream Deniable-Encryption Algorithm Satisfying Criterion of the Computational Indistinguishability from Probabilistic Ciphering

A.A. Moldovyan, D.N. Moldovyan, V.A. Shcherbacov

Abstract

It is proposed a method for stream deniable encryption of secret message, which is computationally indistinguishable from the probabilistic encryption of some fake message. The method uses generation of two key streams with some secure block cipher. One of the key streams is generated depending on the secret key and the other one is generated depending on the fake key. The key streams are mixed with the secret and fake data streams so that the output ciphertext looks like the ciphertext produced by some probabilistic encryption algorithm applied to the fake message with using the fake key. When the receiver or/and sender of the ciphertext are coerced to open the encryption key and the source message, they open the fake key and the fake message. To disclose their lie the coercer should demonstrate possibility of the alternative decryption of the ciphertext, however this is a computationally hard problem.

Keywords: cryptology, algorithm, stream, deniable, encryption

1 Introduction

The notion of deniable encryption (DE) was introduced by Canetti et al. in 1997 [1] as property of cryptographic protocols and algorithms to resist the so called coercive attacks that are performed by some adversary (coercer) that intercepts the ciphertext and has power to force

sender or/and receiver to open both the sent message and the encryption key. If the sender encrypts the secrete message using public key of the receiver of the message, then we have the case of the public-key deniable encryption schemes. If the encryption of the secrete message is performed using a shared secret key, then we have the case of the shared-key deniable encryption schemes.

The public-key DE protocols are applicable for preventing vote buying in the internet-voting systems [2] and for providing security of multiparty computations [3]. The shared-key DE algorithms represent interest for information protection in computer and telecommunication systems. In the literature there are considered sender-deniable [1,2] (coercer attacks the sender of the ciphertext), receiver-deniable [3] (coercer attacks the receiver of the ciphertext), and bi-deniable [4] (coercer attacks the both parties of the secure communication session) cryptoschemes. The encryption scheme is deniable, if it provides possibility to the sender or/and to the receiver to open a fake message and a fake key instead of the secret ones so that disclosing their lie is a computationally infeasible problem for the coercer. Practical methods for bi-deniable public-key encryption have been proposed in [5,6].

Fast methods for block deniable encryption are described in [7]. Those methods implement deniable encryption as simultaneous transformation of two different messages, secret and fake ones, using two keys, secret and fake ones, into the single ciphertext. In the paper [7] it has been also introduced the notion of the computational indistinguishability of the DE from the probabilistic encryption. The DE algorithm is considered as possessing such property, if it produces the ciphertext that can be also produced by some probabilistic-encryption algorithm used for ciphering the fake message with the fake key and some random input. The stream DE algorithms proposed in [7] and [8] are indistinguishable from some probabilistic encryption algorithms, however those algorithms are very slow. At present no practical and fast algorithms for shared-key stream DE are described in the literature, such algorithms are very attractive for practical application to provide information protection in computer and telecommunication systems though.

The present paper proposes a method and algorithm for sufficiently fast stream bi-deniable encryption. Computational indistinguishability from a probabilistic stream encryption is used as a design criterion. The paper is organized as follows. Section 2 presents the design criteria. Section 3 and 4 present method and algorithm for stream bi-deniable encryption, correspondingly. Section 5 discusses the proposed algorithm. Section 6 concludes the paper.

2 Design criteria

For designing a shared-key DE algorithm the following criteria have been used:

- the algorithm should implement the stream encryption;
- the used encryption method should provide possibility of the independent decryption of each symbol of the produced ciphertext; this criterion takes into account possible practical applications in the cloud-computing technologies for processing data contained in encrypted files having large size;
- the method should implement the DE procedure as simultaneous encryption of the secret and fake messages using the secret and fake keys;
- the output ciphertext generated by the algorithm should be computationally indistinguishable from the ciphertext produced by some probabilistic ciphering a fake message with a fake key;
- the algorithm should provide sufficiently high encryption speed;
- the algorithm should provide bi-deniability;
- one should provide possibility of the independent recovering of the secret and fake messages, using secret or fake key, correspondingly.

3 Encryption method

One can consider the text files as sequence of small data blocks having the fixed size, i.e. as sequence of the bit strings with which the symbols are coded. Thus, for encrypting a file or a message it is possible

to apply formally the fast bi-deniable block-encryption method proposed in [7]. To encrypt a secret message $T = (T_1, T_2, \ldots, T_i, \ldots, T_n)$ represented as sequence of the b-bit data blocks T_i ($b = 32, 64, 128,$ or 256) in that method it is supposed to generate a fake message $M = (M_1, M_2, \ldots, M_i, \ldots, M_n)$, where M_i are the b-bit data blocks, having the same size as secret one and then to encrypt simultaneously all pairs of the data blocks T_i and M_i ($i = 1, 2, \ldots, n$) as follows:

1. Using some known secure block cipher with b-bit input data block encrypt the data block M_i into the b-bit block C_{M_i} of intermediate ciphertext in accordance with the formula

$$C_{M_i} = E_K(M_i), \tag{1}$$

where E is the used block cipher; K is the fake key.

2. Encrypt the data block T_i into the b-bit block C_{T_i} of intermediate ciphertext in accordance with the formula

$$C_{T_i} = E_Q(T_i), \tag{2}$$

where Q is the secret key.

3. Compute the ith ($2b$)-bit block of the output ciphertext C_i as ($2b$)-bit binary polynomial satisfying the system of congruences

$$\begin{cases} C_i \equiv C_{M_i} \bmod \mu(x) \\ C_i \equiv C_{T_i} \bmod \lambda(x), \end{cases} \tag{3}$$

where binary polynomial $\mu(x) = 1||\mu'(x)$, $||$ denotes the concatenation operation; $\mu'(x)$ is the binary polynomial, which is given by the right b bits of the fake key K (i.e. the right b bits of the secret key K are interpreted as binary polynomial); binary polynomial $\lambda(x) = 1||\lambda'(x)$; $\lambda'(x)$ is the binary polynomial, which is given by the right b bits of the secret key Q.

In the method described in [7] the keys K and Q are generated as a pair of random bit strings such that polynomials $\mu'(x)$ and $\lambda'(x)$ are mutually irreducible, therefore the last system of congruences has unique solution $C_i < \lambda(x)\mu(x)$ and can be computed as follows:

$$\begin{aligned} C_i = [&C_{M_i}\lambda(x)(\lambda^{-1}(x) \bmod \mu(x)) + \\ &+ C_{T_i}\mu(x)(\mu^{-1}(x) \bmod \lambda(x))] \bmod \mu(x)\lambda(x). \end{aligned}$$

In the case of small values of the data blocks the described method is insecure, for example, in the case of simultaneous encryption of the files $T = (t_1, t_2, \ldots, t_i, \ldots, t_n)$ and $M = (m_1, m_2, \ldots, m_i, \ldots, m_n)$, where t_i and m_i are symbols having size $b \leq 16$ bits. To overcome this problem we propose to modify the key for each value $i = 1, 2, \ldots, n$. Due to such modification it becomes possible to simplify computation of the blocks C_{M_i} and C_{T_i}, if the sequences of the modified values of the fake and secret key are generated in the form of some pseudorandom sequence that is computationally indistinguishable from the uniform random sequence. Besides, we propose to use unique fake and secret key sequences for encryption of each secret message T. Thus, we have come to idea to generate fake (Γ) and secret (Γ') key sequences using the block cipher E in accordance with the following formulas

$$E_K(i||V) \bmod 2^{2b} = (\alpha_i||\beta_i) \text{ and } E_Q(i||V) \bmod 2^{2b} = (\alpha'_i||\beta'_i),$$

where α_i, β_i, α'_i, and β'_i are b-bit strings such that binary polynomials $\mu_i(x) = 1||\beta_i$ and $\lambda_i(x) = 1||\beta'_i$ are mutually irreducible; V is the 64-bit initialization vector generated at random for each encrypted message or file (the value V is not secret, therefore V can be transmitted via insecure channel).

The sequences Γ and Γ' can be written as follows:

$$\Gamma = \{(\alpha_1||\beta_1), (\alpha_2||\beta_2), \ldots, (\alpha_i||\beta_i), \ldots, (\alpha_n||\beta_n)\} \text{ and}$$
$$\Gamma' = \{(\alpha'_1||\beta'_1), (\alpha'_2||\beta'_2), \ldots, (\alpha'_i||\beta'_i), \ldots, (\alpha'_n||\beta'_n)\}.$$

The elements $(\alpha_i||\beta_i)$ and $(\alpha'_i||\beta'_i)$ of these sequences are to be used to encrypt simultaneously the couple of symbols t_i and m_i. Instead of formulas (1) and (2) one can use the following transformation of the ith symbol of the fake and secret messages, respectively:

$$c_{m_i} = m_i \oplus \alpha_i \quad (4)$$
$$c_{t_i} = t_i \oplus \alpha'_i, \quad (5)$$

where $\oplus$ is the XOR operation. The b-bit symbols c_{m_i} and c_{t_i} of the intermediate ciphertext are to be mixed into the single $(2b)$-bit symbol

c_i of the output ciphertext in accordance with the following formula

$$\begin{aligned} c_i = [c_{m_i}\lambda_i(x)(\lambda_i^{-1}(x) \bmod \mu_i(x))+ \\ + c_{t_i}\mu_i(x)(\mu_i^{-1}(x) \bmod \lambda_i(x))] \bmod \mu_i(x)\lambda_i(x), \end{aligned} \tag{6}$$

where $\mu_i(x) = 1||\beta_i$ and $\lambda_i(x) = 1||\beta'$ are mutually irreducible binary polynomials. Formula (6) defines solution of the following system of congruences

$$\begin{cases} c_i \equiv c_{m_i} \bmod \mu_i(x) \\ c_i \equiv c_{t_i} \bmod \lambda_i(x). \end{cases} \tag{7}$$

System (7) defines the following formulas for computing the symbols c_{m_i} and c_{t_i} from c_i:

$$c_{m_i} = c_i \bmod \mu_i(x), \tag{8}$$

$$c_{t_i} = c_i \bmod \lambda_i(x). \tag{9}$$

Then the ith symbols t_i and m_i of the source texts T and M are computed using the values α_i and α'_i with the following formulas ($i = 1, 2, \ldots, n$):

$$m_i = c_{m_i} \oplus \alpha_i, \tag{10}$$

$$t_i = c_{t_i} \oplus \alpha'_i. \tag{11}$$

4 The stream deniable encryption algorithm

Suppose we have a secure block cipher E with 128-bit input data block and 128-bit key K. Using the method described in Section 3 (in which it is supposed two parties of the communication session share the secret 128-bit key Q and the fake 128-bit key K) we have constructed the following algorithm for performing the stream DE of the secret message T:

INPUT: the secret message $T = (t_1, t_2, \ldots, t_i, \ldots, t_n)$ and encryption keys K and Q.

1. Generate a fake message M having the same length as the message T.

2. Generate a random value of the 64-bit initialization vector V.

3. For $i = 1$ to n do the following steps.

3.1. Using the procedure **Form_**$\alpha\beta$ generate the ith elements $(\alpha_i||\beta_i)$ and $(\alpha'_i||\beta'_i)$ of the key sequences Γ and Γ'.

3.2. Compute the b-bit symbols c_{m_i} and c_{t_i} of the intermediate ciphertext using formulas (4) and (5).

3.3. Compute the $(2b)$-bit symbol c_i of the output ciphertext as solution of the system of two linear congruences (7), which is defined by formula (6).

4. Compose the output ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$.

OUTPUT: the ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$ and the initialization vector V.

The procedure **Form_**$\alpha\beta$ used at step 3 is described as follows:

INPUT: two 128-bit keys K and Q and two 64-bit values i and V.

1. Compute the value $(\alpha_i||\beta_i) = E_K(i||V) \bmod 2^{2b}$, where E is some specified 128-bit block cipher; α_i and β_i are b-bit strings; the value $E_K(i||V)$ is considered as binary number.

2. Compose the bit string $\mu_i = (1||\beta_i)$.

3. Compute the value $(\alpha'_i||\beta'_i) = E_Q(i||V) \bmod 2^{2b}$.

4. Compose the bit string $\lambda_i = (1||\beta'_i)$.

5. Considering the bit strings μ_i and λ_i as binary polynomials $\mu_i(x)$ and $\lambda_i(x)$ of the degree b, respectively, compute the greatest common divisor $D = gcd(\mu_i(x), \lambda_i(x))$.

6. If $D \neq 1$, then increment $\beta'_i \leftarrow \beta'_i + 1 \bmod 2^b$ (here the bit string β'_i is considered as binary number) and go to step 4, otherwise STOP.

OUTPUT: two $(2b)$-bit elements $(\alpha_i||\beta_i)$ and $(\alpha'_i||\beta'_i)$ of the key sequences Γ and Γ'.

Decryption of the ciphertext C produced by the proposed DE algorithm requires using the value V assigned to C (i.e. sent together with the ciphertext C) and both the secret and fake keys and is to be performed with the following algorithm.

Algorithm for decrypting the secret message.

INPUT: the ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$, the encryption key Q, the fake key K, and the initialization vector V.

1 For $i = 1$ to n do the following steps.

1.1. Using the procedure **Form_**$\alpha\beta$ generate the ith element $(\alpha'_i||\beta'_i)$ of the key sequence Γ'.

1.2. Compute the b-bit symbol c_{t_i} of the intermediate ciphertext using the formula (9).

1.3. Compute the b-bit symbol t_i of the secret message using formula (11).

2. Compose the message $T = (t_1, t_2, \ldots, t_i, \ldots, t_n)$.

OUTPUT: the opened message T.

5 Discussion

5.1 Security against the two-side coercive attack

Suppose a coercive adversary intercepts the ciphertext and initialization vector sent by sender to receiver of secret message and then forces both the parties to open the message, the encryption and decryption algorithms, and the encryption key. The encryption algorithm proposed in Section 4 resists this attack, since the sender and the receiver are able to fulfill coercers demands without opening the secret message. For this purpose they open the following

- the fake key K declared as the secret one;

- the fake message M declared as the secret one;

- probabilistic encryption algorithm that allegedly produced the ciphertext intercepted by the coercer;

- decryption algorithm that discloses the fake message from the cryptogram, while using the fake key.

To catch them in a lie, the coercer should show conclusively that the ciphertext contains another message. The last can be performed with guessing the secret key Q, however this method is impractical due to sufficiently large size of the value Q (128 bits).

Let us also consider the known-plaintext attack, i.e. suppose the coercer knows the secret message. If he will be able to compute the secret key Q, then he will be able to prove the sender and the receiver are cheating (the proving consists in opening the message T from the ciphertext C, while using the key Q). Suppose additionally that, using

the known message T and the value V, the coercer is able to compute the key sequence Γ' and then all values $E_K(i||V)$, where $i = 1, 2, \ldots, n$ (see step 3 in description of the procedure **Form_**$\alpha\beta$).

In this case the assumption about possibility to compute the key Q from the known 128-bit input $i||V$ and output values $E_K(i||V)$ leads to conclusion about insecurity of the used block cipher E against the known-plaintext attack. However in the proposed DE algorithm it is used a secure block cipher, for example, AES that surely resists such attacks and is recommended by the standard ISO/IET 18033-3:2010 [9].

Thus, one can conclude the proposed DE algorithm provides bi-deniability. The probabilistic encryption algorithm to be opened to the coercer is described as follows.

Associated probabilistic stream encryption algorithm

INPUT: the message $M = (m_1, m_2, \ldots, m_i, \ldots, m_n)$ and the encryption key K.

1. Generate a random value of the 64-bit initialization vector V.

2. For $i = 1$ to n do the following steps.

2.1. Compute the value $(\alpha_i||\beta_i) = E_K(i||V) \bmod 2^{2b}$, where E is the specified 128-bit block cipher; α_i and β_i are b-bit strings; the value $E_K(i||V)$ is considered as binary number.

2.2. Compose the bit string $\mu_i = (1||\beta_i)$.

2.3. Generate randomly two b-bit strings ρ and η'.

2.4. Compose the bit string $\eta = (1||\eta')$.

2.5. Considering the bit strings μ_i and η as binary polynomials $\mu_i(x)$ and $\eta(x)$ of the degree b, respectively, compute the greatest common divisor $D = gcd(\mu_i(x), \eta(x))$.

2.6. If $D \neq 1$, then increment $\eta' \leftarrow \eta' + 1 \bmod 2^b$ (here the bit string η' is considered as binary number) and go to step 2.4, otherwise go to step 2.7.

2.7. Compute the b-bit symbol c_{m_i} of the intermediate ciphertext using formula (4).

2.8. Compute the $(2b)$-bit symbol c_i as solution of the following system of two linear congruences:

$$\begin{cases} c_i \equiv c_{m_i} \bmod \mu_i(x) \\ c_i \equiv \rho(x) \bmod \eta(x), \end{cases} \tag{12}$$

where the bit string c_{m_i} is considered as binary polynomial and $\rho(x)$ is the binary polynomial represented by the bit string ρ.

3. Compose the output ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$.
OUTPUT: the ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$ and the initialization vector V.

The value c_i at step 2.8 can be computed using the following formula:

$$\begin{aligned} c_i = [c_{m_i}\eta(x)(\eta^{-1}(x) \bmod \mu_i(x)) + \\ + \rho(x)\mu_i(x)(\mu_i^{-1}(x) \bmod \eta(x))] \bmod \mu_i(x)\eta(x). \end{aligned}$$

It is easy to see that for each symbol c_i of the ciphertext C there exist different bit strings η' and ρ satisfying system (12). Indeed, for given c_i and arbitrary η' such that $gcd(\mu_i(x), \eta(x)) = 1$ the value ρ satisfying (12) can be computed as binary polynomial $\rho(x) = c_i \bmod \eta(x)$, where the bit string c_i is considered as binary polynomial.

Thus, while using the encryption key K the associated probabilistic encryption algorithm can potentially encrypt the message M into the cryptogram C produced by the DE algorithm. Since it is computationally difficult to prove that the ciphertext C was produced by the DE process, but not by the probabilistic encryption, one can say the proposed DE algorithm is computationally indistinguishable from the associated probabilistic encryption algorithm.

The decryption algorithm to be opened to the coercer is described as follows.

Dishonest decryption algorithm

INPUT: the ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_n)$, the encryption key K, and the initialization vector V.

1 For $i = 1$ to n do the following steps.

1.1. Compute the value $(\alpha_i||\beta_i) = E_K(i||V) \bmod 2^{2b}$.

1.2. Compose the bit string $\mu_i = (1||\beta_i)$.

1.3. Compute the b-bit symbol c_{m_i} of the intermediate ciphertext using the formula (8).

1.4. Compute the b-bit symbol m_i using the formula (10).

2. Compose the message $M = (m_1, m_2, \ldots, m_i, \ldots, m_n)$.

OUTPUT: the opened message M.

5.2 Estimation of the encryption speed

For comparing the performance of the proposed algorithm with the stream DE algorithm described in [7] one can roughly assume that time complexity of computation of the value c_i in accordance with the formula (6) is equal to the time complexity of one block encryption operation. Besides, the time complexity of generation of the values $(\alpha_i||\beta_i) = E_K(i||V) \bmod 2^{2b}$ and $(\alpha'_i||\beta'_i) = E_Q(i||V) \bmod 2^{2b}$ is approximately equal to 1 and 2 block-encryption operations, correspondingly.

Thus, the time complexity of the encryption of one symbol of the secret message is equal to ≈ 4 block-encryption operations. Taking the last into account one can get the following formula for the encryption speed of the proposed algorithm:

$$S = \frac{1}{4} \cdot \frac{b}{128} S_E, \tag{13}$$

where b is the bit length of the symbols with which the secret message is written; S_E is the encryption speed of the block cipher E.

While implementing the DE method from [7] with using the block cipher E, encryption of one symbol of the secret message takes on the average 2^{2b+1} operations of the block encryption and defines the following formula for estimating the speed

$$S_{[7]} = \frac{b}{128} \cdot \frac{S_E}{2^{2b+1}}. \tag{14}$$

Comparing (13) with (14) one can state that the proposed stream DE algorithm is significantly faster (by 2^{2b-1} times) than algorithm by method [7]. For example, in the case $b = 8$ the ratio S/S_E is equal to 2^{15}.

6 Conclusion

It is proposed a method and algorithm for fast stream deniable encryption satisfying criterion of the computational indistinguishability from the stream probabilistic encryption. It has been shown that the DE algorithm resists two-side coercive attack. As compared with the stream DE algorithm presented in [7] the proposed one is significantly faster, the algorithm from [7] has one interesting advantage though. The advantage consists in using the same decryption algorithm for opening both the secret and the fake messages from the ciphertext. Such property is significant for providing security against coercive attacks combined with measuring duration of the decryption process. In our future research we plan to develop a fast stream DE method with the same algorithm that decrypts the secret and fake message.

References

[1] R. Canetti, C. Dwork, M. Naor, R. Ostrovsky. *Deniable Encryption*. Proceedings Advances in Cryptology – CRYPTO 1997. Lectute Notes in Computer Science. Springer - Verlag. Berlin, Heidelberg, New York, 1997. Vol. 1294. pp. 90–104.

[2] J. Howlader, S. Basu. *Sender-Side Public Key Deniable Encryption Scheme.* Advances in Recent Technologies in Communication and Computing. Proceedings of the ARTCom '09 International Conference. 2009. pp. 9-13. (DOI: 10.1109/ARTCom.2009.107)

[3] Bo Meng, Jiang Qing Wang. *A Receiver Deniable Encryption Scheme*. Proceedings of the 2009 International Symposium on Information Processing (ISIP09). Huangshan, P. R. China, August 21-23, 2009. pp. 254–257.

[4] A. O'Neil, C. Peikert, B. Waters. *Bi-Deniable Public-Key Encryption*. Advances in Cryptology CRYPTO 2011. Lectute Notes in

Computer Science. Springer Verlag. Berlin, Heidelberg, New York, 2011. Vol. 6841. pp. 525–542.

[5] A.A. Moldovyan, N.A. Moldovyan. *Practical Method for Bi-Deniable Public-Key Encryption*. Quasigroups and related systems. 2014. Vol. 22. P. 277–282.

[6] A. A. Moldovyan, N. A. Moldovyan, V. A. Shcherbacov. *Bi-Deniable Public-Key Encryption Protocol Secure Against Active Coercive Adversary*. Buletinul Academiei de Stiinte a Republicii Moldova. Matematica. 2014. N. 3 (76). P. 23–29.

[7] E.V. Morozova, Ya.A. Mondikova, N.A.Moldovyan. *Methods for shared-key deniable encryption*. Informatsionno-upravliaiushchie sistemy, 2013, no. 6, pp. 73–78 (in Russian).

[8] N.A. Moldovyan, A. R. Birichevskiy, Ya.A. Mondikova. *Deniable Encryption based on block ciphers*. Informatsionno-upravliaiushchie sistemy, 2014, no. 5, pp. 80–86 (in Russian).

[9] *International standard ISO/IEC 18033-3:2010*. Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers.

A. A. Moldovyan[1] , D. N. Moldovyan[2],
V. A. Shcherbacov[3],

Received July 10, 2015

[1] Professor, ITMO University
Kronverksky pr., 10, St.Petersburg, 197101
Russia
E–mail: maa1305@yandex.ru

[2] Dr., St. Petersburg Institute for Informatics and Automation of
Russian Academy of Sciences
14 Liniya, 39, St.Petersburg, 199178
Russia E–mail: mdn.spectr@mail.ru

[3] Dr., Institute of Mathematics and Computer Science
Academy of Sciences of Moldova Academiei str. 5, MD−2028 Chişinău
Moldova
E–mail: scerb@math.md

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

On some applications of quasigroups in cryptography

N.A. Moldovyan A.V. Shcherbacov V.A. Shcherbacov

Abstract

In the paper we present based on quasigroups new deniable encryption method, relatively fast stream cipher and generalisation of El-Gamal scheme.

Keywords: cryptology, quasigroup, algorithm, stream, deniable, encryption, El-Gamal scheme

1 Deniable-encryption mode for block ciphers

Deniable encryption (DE) is a method for generating ciphertexts that can be alternatively decrypted providing security against the so called coercive attacks [3] for which it is assumed that after ciphertext has been sent the adversary has possibility to force both the sender and the receiver to open the plaintext corresponding to the ciphertext and the encryption key. In the case of block ciphering the DE can be provided with simultaneous encryption of the secret and fake messages using the secret and fake keys, correspondingly. While being coerced the sender and receiver of the ciphertext open the fake key and fake message and declare they have used the probabilistic encryption [4]. Earlier in paper [5] it had been proposed a method for simultaneous encryption of two messages based on solving a system of two linear equations. In this section we propose design of the DE mode for using block ciphers, which is based on the mentioned method.

Definition 1. *Binary groupoid* $(G, \circ)$ *is isotopic image of a binary groupoid* $(G, \cdot)$*, if there exist permutations* α, β, γ *of the set* G *such that* $x \circ y = \gamma^{-1}(\alpha x \cdot \beta y)$ *[1].*

Suppose E_V be a block encryption algorithm with n-bit input data block and the value used as encryption key. All existing n-bit data blocks can be considered as elements of some quasigroup with the operation $*$ defined as follows:

$$K * i = E_V(K \oplus E_V(i)),$$

where $\oplus$ is the XOR operartion; K and i are n-bit vectors. This quasigroup is isotope of the group $(G, \oplus)$, where G is the set of all n-bit vectors. Here E_V is a permutation of the symmetric group S_G.

Evidently, for all possible values i and $Q \neq K$ we have

$$E_V(Q \oplus E_V(i)) \neq E_V(K \oplus E_V(i)). \tag{1}$$

Using this property of the quasigroup one can define simultaneous encryption of two different messages $T = (t_1, t_2, \ldots, t_i, \ldots, t_z)$ and $M = (m_1, m_2, \ldots, m_i, \ldots, m_z)$, where $z < 2^n$; t_i and m_i are n-bit data blocks, as generation of the single ciphertext $C = (c_1, c_2, \ldots, c_i, \ldots, c_z)$ containing $(2n)$-bit ciphertext blocks $c_i = (c'_i, c''_i)$, where c'_i and c''_i are n-bit values, computed from the following system of equations in the field $GF(2^n)$:

$$\begin{cases} c'_i + A_i c''_i \equiv B_i + m_i \bmod \eta(x) \\ c'_i + G_i c''_i \equiv H_i + t_i \bmod \eta(x), \end{cases} \tag{2}$$

where $\eta(x)$ is some specified irreducible binary polynomial of the degree n; the n-bit values A_i, B_i, G_i, and H_i are computed using the random n-bit initialization vector V (this value is not secret) as follows:

$$A_i = E_V(K \oplus E_V(i)); G_i = E_V(Q \oplus E_V(i));$$
$$B_i = E_K(A_i); H_i = E_Q(G_i).$$

While solving (1) the values A_i, B_i, G_i, and H_i are considered as binary polynomials of the degree $s < n$. Due to condition (1) the system (2) always has the single solution, therefore the proposed deniable-encryption procedure is defined correctly. Let us agree that the secret message (key) is the value T (Q) and the fake message (key) is the value M (K). If the coercer forces the sender and receiver of the secret message T to open the ciphertext C and the encryption key, then

they open the fake key K and the fake message M and declare using the probabilistic block-encryption mode implemented with the block cipher E. In terms of paper [4] the declared encryption algorithm is called the associated encryption algorithm.

In the case of the proposed deniable-encryption method the last algorithm is described as consecutive probabilistic encryption of the data blocks m_i for each value $i = 1, 2, \ldots, z$ performing the following steps:

1. Generate a random initialization vector V and compute the values $A_i = E_V(K \oplus E_V(i))$ and $B_i = E_K(A_i)$.

2. Generate a random binary polynomial $\rho_i(x)$ of the degree $s < n$.

3. Compute the unknowns c_i' and c_i'' from the following system of equations in $GF(2^n)$:

$$\begin{cases} c_i' + A_i c_i'' \equiv B_i + m_i \bmod \eta(x) \\ c_i' + \rho_i c_i'' \equiv 1 \bmod \eta(x), \end{cases} \tag{3}$$

Evidently, for some sequence of the values $\rho_1(x), \rho_2(x), \ldots, \rho_z(x)$ the message M is transformed with the key K into the given ciphertext C.

To distinguish the use of the deniable encryption with the system (2) from the probabilistic encryption with the system (3) the potential coercive attacker should compute the key Q. The last problem is computationally difficult, if E is a secure block cipher, for example, AES [7] with 128-bit key and $n = 128$. Restoring the secret message from the ciphertext is performed as decryption of each ciphertext block $c_i = (c_i', c_i')$, $i = 1, 2, \ldots, z$, as follows:

1. Using the secret key Q compute the values $G_i = E_V(Q \oplus E_V(i))$ and $H_i = E_Q(G_i)$.

2. Compute the plaintext data block $t_i = c_i' + G_i c_i'' - H_i \bmod \eta(x)$.

The fake decryption of the ciphertext is as follows ($i = 1, 2, \ldots, z$):

1. Using the fake key K compute the values $A_i = E_V(K \oplus E_V(i))$ and $B_i = E_K(A_i)$.

2. Compute the plaintext data block $m_i = c_i' + A_i c_i'' - B_i \bmod \eta(x)$.

2 Stream cipher on base of binary quasigroups

Here we give more detailed description of algorithm which was proposed in [10]. This algorithm simultaneously uses two cryptographical procedures: enciphering using generalisation of Markovski stream algorithm [11] and enciphering using a system of orthogonal operations.

We also give some realisation of this algorithm on base of T-quasigroups, more precise, on the base of medial quasigroups. Necessary information about quasigroups and some its applications in cryptography can be found in [1, 8, 10].

Below we denote the action of the left (right, middle) translation in the power a of a binary quasigroup (Q, g_1) on the element u_1 by the symbol ${}_{g_1}T^a_{l_1}(u_1)$. And so on. Here l_1 means leader element. See [8, 10, 11] for details.

Algorithm 1. *Enciphering. Initially we have plaintext* $u_1, u_2, \ldots, u_6$.

$$
\begin{aligned}
&\textit{Step } 1.\\
&{}_{g_1}T^a_{l_1}(u_1) = v_1\\
&{}_{g_2}T^b_{l_2}(u_2) = v_2\\
&F^c_1(v_1, v_2) = (v'_1, v'_2)\\
&\textit{Step } 2.\\
&{}_{g_3}T^d_{v'_1}(u_3) = v_3\\
&{}_{g_4}T^e_{v'_2}(u_4) = v_4\\
&F^f_2(v_3, v_4) = (v'_3, v'_4)\\
&\textit{Step } 3.\\
&{}_{g_5}T^g_{v'_3}(u_5) = v_5\\
&{}_{g_6}T^h_{v'_4}(u_6) = v_6\\
&F^i_3(v_5, v_6) = (v'_5, v'_6).
\end{aligned}
\tag{4}
$$

We obtain ciphertext $v'_1, v'_2, \ldots, v'_6$.

Deciphering. Initially we have ciphertext $v'_1, v'_2, \ldots, v'_6$.

$$
\begin{array}{l}
\textit{Step } 1. \\
F_1^{-c}(v'_1, v'_2) = (v_1, v_2) \\
{}_{g_1}T_{l_1}^{-a}(v_1) = u_1 \\
{}_{g_2}T_{l_2}^{-b}(v_2) = u_2 \\
\textit{Step } 2. \\
F_2^{-f}(v'_3, v'_4) = (v_3, v_4) \\
{}_{g_3}T_{v'_1}^{-d}(v_3) = u_3 \\
{}_{g_4}T_{v'_2}^{-e}(v_4) = u_4 \\
\textit{Step } 3. \\
F_3^{-i}(v'_5, v'_6) = (v_5, v_6) \\
{}_{g_5}T_{v'_3}^{-g}(v_5) = u_5 \\
{}_{g_6}T_{v'_4}^{-h}(v_6) = u_6
\end{array}
\tag{5}
$$

We obtain plaintext $u_1, u_2, \ldots, u_6$.

From Algorithm 1 we obtain classical Markovski algorithm, if we take only one quasigroup, one kind of quasigroup translations (left translations) any of which is taken in power = 1, and, finally, if system of orthogonal operations (crypto-procedure F) is not used. Some generalisations of Algorithm 1 are given in [12].

3 T-quasigroup based stream cipher

We give a numerical example of encryption Algorithm 1 based on T-quasigroups (more exactly, on medial quasigroups) [12]. Notice that the number 257 is prime. Form of parastrophes of T-quasigroups, for example, of quasigroup $(A, \overset{(13)}{*})$ can be found in [12], [6, p. 39].

Example 1. *Take the cyclic group* $(Z_{257}, +) = (A, +)$.

1. *Define T-quasigroup* $(A, *)$ *with the form* $x * y = 2 \cdot x + 131 \cdot y + 3$ *with a leader element* l*, say,* $l = 17$*. Denote the mapping* $x \mapsto x * l$ *by the letter* g_1*, i.e.* $g_1(x) = x * l$ *for all* $x \in A$*.*

   In order to find the mapping g_1^{-1} *we find the form of operation* $\overset{(13)}{*}$*. We have* $x \overset{(13)}{*} y = 129 \cdot x + 63 \cdot y + 127$*,* $f^{-1}x = x \overset{(13)}{*} l$*. Then* $g_1^{-1}(g_1(x)) = g_1^{-1}(x * l) = (x * l) \overset{(13)}{*} l = x$*.*

   In some sense quasigroup $(A, \overset{(13)}{*})$ *is the "right inverse quasigroup" to quasigroup* $(A, *)$*. Notice that from results of article [6, Theorem 16] it follows that* $(A, *) \perp (A, \overset{(13)}{*})$*.*

2. *Define T-quasigroup* $(A, \circ)$ *with the form* $x \circ y = 10 \cdot x + 81 \cdot y + 53$ *with a leader element* l*, say,* $l = 71$*. Denote the mapping* $x \mapsto l * x$ *by the letter* g_2*, i.e.* $g_2(x) = l \circ x$ *for all* $x \in A$*.*

   In order to find the mapping g_2^{-1} *we find the form of operation* $\overset{(23)}{\circ}$*. We have* $x \overset{(23)}{\circ} y = 149 \cdot x + 165 \cdot y + 250$*.*

3. *Define a system of two parastroph orthogonal T-quasigroups* $(A, \cdot)$ *and* $(A, \overset{(23)}{\cdot})$ *in the following way*

$$\begin{cases} x \cdot y = 3 \cdot x + 5 \cdot y + 6 \\ x \overset{(23)}{\cdot} y = 205 \cdot x + 103 \cdot y + 153. \end{cases}$$

   Denote quasigroup system $(A, \cdot, \overset{(23)}{\cdot})$ *by* $F(x, y)$*, since this system is a function of two variables.*

   In order to find the mapping $F^{-1}(x, y)$ *we solve the system of linear equations*

$$\begin{cases} 3 \cdot x + 5 \cdot y + 6 = a \\ 205 \cdot x + 103 \cdot y + 153 = b. \end{cases}$$

   We have $\Delta = 55$*,* $1/\Delta = 243$*,* $x = 100 \cdot a + 70 \cdot b + 255$*,* $y = 43 \cdot a + 215 \cdot b$*. Therefore we have, if* $F(x, y) = (a, b)$*, then* $F^{-1}(a, b) =$

$(100 \cdot a + 70 \cdot b + 255, 43 \cdot a + 215 \cdot b)$, *i.e.*

$$\begin{cases} x = 100 \cdot a + 70 \cdot b + 255 \\ y = 43 \cdot a + 215 \cdot b. \end{cases}$$

We have defined the mappings g_1, g_2, F *and now we can use them in Algorithm 1.*

Let 212; 17; 65; 117 *be a plaintext. We take the following values in formula (4):* $a = b = d = e = f = 1; c = 2$. *Below we use Gothic font to distinguish leader elements, i.e., the numbers* $\mathfrak{17}$ *and* $\mathfrak{71}$ *are leader elements. Then*

Step 1.

$g_1(212) = 212 * \mathfrak{17} = 2 \cdot 212 + 131 \cdot 17 + 3 = 84$

$g_2(17) = \mathfrak{71} \circ 17 = 10 \cdot 71 + 81 \cdot 17 + 53 = 84$

$F(84; 84) = (3 \cdot 84 + 5 \cdot 84 + 6; 205 \cdot 84 + 103 \cdot 84 + 153) = (164; 68)$

$F(164; 68) = (3 \cdot 164 + 5 \cdot 68 + 6; 205 \cdot 164 + 103 \cdot 68 + 153) =$ ***(67; 171)***

Step 2.

$g_1(65) = 65 * 67 = 2 \cdot 65 + 131 \cdot 67 + 3 = 172$

$g_2(117) = 171 \circ 117 = 10 \cdot 171 + 81 \cdot 117 + 53 = 189$

$F(172; 189) = (3 \cdot 172 + 5 \cdot 189 + 6; 205 \cdot 172 + 103 \cdot 189 + 153) =$ ***(182; 139)***

We obtain the following ciphertext 67; 171; 182; 139.

For deciphering we use formula (5).

Step 1.

$F^{-1}(67; 171) = (100 \cdot 67 + 70 \cdot 171 + 255, 43 \cdot 67 + 215 \cdot 171) = (164; 68)$

$F^{-1}(164; 68) = (100 \cdot 164 + 70 \cdot 68 + 255, 43 \cdot 164 + 215 \cdot 68) = (84; 84)$

$g_1^{-1}(84) = 84 \overset{(13)}{*} 17 = 129 \cdot 84 + 63 \cdot 17 + 127 =$ ***212***

$g_2^{-1}(84) = 71 \overset{(23)}{\circ} 84 = 149 \cdot 71 + 165 \cdot 84 + 250 =$ ***17***

Step 2.

$F^{-1}(182; 139) = (100 \cdot 182 + 70 \cdot 139 + 255, 43 \cdot 182 + 215 \cdot 139) = (172; 189)$

$g_1^{-1}(172) = 172 \overset{(13)}{*} 67 = 129 \cdot 172 + 63 \cdot 67 + 127 =$ ***65***

$$g_2^{-1}(189) = 171 \overset{(23)}{\circ} 189 = 149 \cdot 171 + 165 \cdot 189 + 250 = \boldsymbol{117}$$

A program using freeware version of programming language Pascal was developed. Experiments demonstrate that encoding-decoding is executed sufficiently fast.

Remark 1. *Proper binary groupoids are more preferable than linear quasigroups by construction of the mapping $F(x, y)$ in order to make encryption more safe, but in this case decryption may be slower than in linear quasigroup case and definition of these groupoids needs more computer (or some other device) memory. The same remark is true for the choice of the function g. Maybe a golden mean in this choice problem is to use linear quasigroups over non-abelian, especially simple, groups.*

Remark 2. *In this cipher there exists a possibility of protection against standard statistical attack. For this scope it is possible to denote more often used letters or pair of letters by more than one integer or by more than one pair of integers.*

4 De-symmetrisation of Markovski algorithm

We give an analogue of El Gamal encryption system based on Markovski algorithm.

Let (Q, f) be a binary quasigroup and $T = (\alpha, \beta, \gamma)$ be its isotopy.

Alices keys are as follows:

Public Key is (Q, f), T, $T^{(m,n,k)} = (\alpha^m, \beta^n, \gamma^k)$, $m, n, k \in \mathbb{N}$, and Markovski algorithm.

Private Key m, n, k.

Encryption

To send a message $b \in (Q, f)$ Bob computes $T^{(r,s,t)}$, $T^{(mr,ns,kt)}$ for a random $r, s, t \in \mathbb{N}$ and $(T^{(mr,ns,kt)}(Q, f))$.

The ciphertext is $(T^{(r,s,t)}, T^{(mr,ns,kt)}(Q, f), (T^{(mr,ns,kt)}(Q, f))b)$.

To obtain $(T^{(mr,ns,kt)}(Q, f))b$ Bob uses Markovski algorithm which is known to Alice.

Decryption

Alice knows m, n, k, so if she receives the ciphertext

$$(T^{(r,s,t)}, T^{(mr,ns,kt)}(Q,f), (T^{(mr,ns,kt)}(Q,f))b),$$

she computes $T^{(-rm,-ns,-kt)}$ from $T^{(r,s,t)}$ and then (Q,f), further she computes $(Q,f)^{-1}$ and, finally, she computes b.

In this algorithm it can also be used isostrophy [9] instead of isotopy, Algorithm 1 instead of Markovski algorithm, n-ary ($n > 2$) quasigroups [2, 10] instead of binary quasigroups.

References

[1] V.D. Belousov. *Foundations of the Theory of Quasigroups and Loops.* Nauka, Moscow, 1967. (in Russian).

[2] V.D. Belousov. *n-Ary Quasigroups.* Stiintsa, Kishinev, 1971. (in Russian).

[3] R. Canetti, C. Dwork, M. Naor, R.Ostrovsky. *Deniable Encryption.* Proceedings Advances in Cryptology CRYPTO 1997, Lectute Notes in Computer Science, 1294:90–104, 1997.

[4] A.A. Moldovyan, N.A. Moldovyan. *Practical method for bi-deniable public-key encryption.* Quasigroups and related systems, 22:277–282, 2014.

[5] A.A. Moldovyan, N.A. Moldovyan, V. A. Shcherbacov. *Bi-deniable public-key encryption protocol secure against active coercive adversary.* Buletinul Academiei de Stiinte a Republicii Moldova. Matematica, (3):23–29, 2014.

[6] G.L. Mullen, V.A. Shcherbacov. *On orthogonality of binary operations and squares.* Bul. Acad. Stiinte Repub. Mold., Mat., (2):3–42, 2005.

[7] J. Pieprzyk, Th. Hardjono, J. Seberry. *Fundumentals of Computer Security.* Springer-Verlag, Berlin, 2003.

[8] V.A. Shcherbacov. *Elements of quasigroup theory and some its applications in code theory,* 2003. links: www.karlin.mff.cuni.cz/drapal/speccurs.pdf; http://de.wikipedia.org/wiki/Quasigruppe.

[9] V.A. Shcherbacov. *On the structure of left and right F-, SM- and E-quasigroups.* J. Gen. Lie Theory Appl., 3(3):197–259, 2009.

[10] V.A. Shcherbacov. *Quasigroups in cryptology.* Comput. Sci. J. Moldova, 17(2):193–228, 2009.

[11] V.A. Shcherbacov, N.A. Moldovyan. *About one cryptoalgorithm.* In Proceedings of the Third Conference of Mathematical Society of the Republic of Moldova dedicated to the 50th anniversary of the foundation of the Institute of Mathematics and Computer Science, August 19-23, 2014, Chisinau, pp. 158–161, Chisinau, 2014. Institute of Mathematics and Computer Science.

[12] Victor Shcherbacov. *Quasigroup based crypto-algorithms,* 2012. arXiv:1201.3016.

N. A. Moldovyan[1], A. V. Shcherbacov[2],
V. A. Shcherbacov[3],

Received July 15, 2015

[1] Professor, St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences
14 Liniya, 39, St.Petersburg, 199178
Russia
E–mail: nmold@mail.ru

[2] M.Sc., Theoretical Lyceum "C. Sibirschi"
Lech Kaczyski str. 4, MD-2028, Chişinău
Moldova
E–mail: admin@sibirsky.org

[3] Dr., Institute of Mathematics and Computer Science
Academy of Sciences of Moldova Academiei str. 5, MD−2028 Chişinău
Moldova
Email: scerb@math.md

Part 10

Databases, artificial intelligence

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Computer Simulation of Multi-optional Decisions

Ion Bolun, Alexandru Costas

Abstract

The formula for the estimation of average disproportion of seats allocation using Hamilton method is obtained. It is shown that, by proportionality of voters' will representation in the final multi-optional decision, the best from the d'Hondt, Huntington-Hill and Sainte-Laguë methods, is the last one. Also, the use of Sainte-Laguë method is easier than that of complemented Webster method. Moreover, the proposed monotone adapted Sainte-Laguë method is considerably better than the Huntington-Hill one. So, for apportionment in the United States Congress House of Representatives, the adapted Sainte-Laguë method is more convenient than the used from 1941 year Huntington-Hill method.

Keywords: disproportion, votes-decision methods, computer simulation, comparison, predicting disproportionality

1 Introduction

The main issue of multi-optional decision-making systems with proportional representation (PR) is the disproportion of voters' will representation in final decision. As criteria of disproportionality it is opportune to use the Average relative deviation index (ARD) I_d [2]. Its minimum value is ensured by Hamilton (Hare) method [1, 4]. However, this method is not immune to the Alabama, of Population and of New State paradoxes [1]. Therefore, in many cases they deny its application in the benefit of monotonous divisor methods, such as d'Hondt, Sainte-Laguë and Huntington-Hill ones [1, 4]. At the same time, it is

not strictly determined which of these "votes-decisions" (VD) methods is more convenient.

Qualitative comparison of Hamilton, Huntington-Hill, d'Hondt, Sainte-Laguë and Mixed VD methods by disproportionality (I_d), quota rule, immunity to paradoxes and non-favoring parties, basing on results from [1, 3-5], are systemized in [7]. Quantitative comparisons of these five methods for particular cases, by average disproportion and non-favoring parties, were done in [1, 3, etc.]. Some results of comparison of Hamilton, d'Hondt and Mixed VD methods by computer simulation are described in [7].

Known results of comparing VD methods are extended, in this paper, by computer simulation, using the elaborated application SIMOD. The average value of I_d index for optimal solutions using Huntington-Hill and the proposed adapted Sainte-Laguë methods are added to the existing results. The comparative analyses of monotone methods with divisor are also done. The obtained results would allow the argued choose of appropriate VD method. A case study in this aim is described.

There are also compared the theoretically obtained mathematical expressions on the average value of I_d index for optimal solutions using Hamilton method with results obtained by computer simulation.

The most known practices with refer to multi-optional decisions are, probably, the ones related to elections. Therefore, further, the addressed aspects of multi-optional decision-making systems will be investigated (not harming the universality) through party-lists elections.

2 The optimization problem

Let [8]: M – number of seats in the elective body; n – number of parties that have reached or exceeded the representation threshold; V – total valid votes cast for the n parties; $d = M/V$ – influence power (rights) of each elector (decider); V_i, v_i – number and, respectively, percentage of valid votes cast for party i; x_i, m_i – number and, respectively, percentage of seats to be allocated to party i; I_d – value of ARD index. Here $V_1 + V_2 + V_3 + \ldots + V_n = V$.

Knowing quantities (integers): M; n; V_i, $i = \overline{1,n}$, it is required to determine the nonnegative values of unknowns x_i ($i = \overline{1,n}$) – integers, which would ensure the minimization of the index I_d value

$$I_d = \sum_{i=1}^{n} |v_i - m_i| \to \min \tag{1}$$

in compliance with the restriction

$$\sum_{i=1}^{n} x_i = M. \tag{2}$$

Problem (1)-(2) is of mathematical programming in integers. The minimum value I_d^* of ARD index is obtained using Hamilton method [1, 4].

3 Disproportionality of Hamilton method's solutions

In this section, results of mathematical expectancy $\bar{I}_d^*$ of I_d^* values, obtained by simulation, are compared with the theoretical approximate.

3.1 Theoretical mathematical expectancy of disproportionality

In comparative analyses, but also for various forecasts, the definition domain of I_d^* values is of interest. Knowledge of definition domain and, also, of mathematical expectancy $\bar{I}_d^*$ expands the information about the possible I_d^* values in concrete elections in the tendency to minimize the index I_d value. This domain is determined in [9]. The knowledge of mathematical expectancy $\bar{I}_d^*$ of I_d^* values, theoretically investigated in [10] would also extend the possibilities of the mentioned above analyses.

Using I_d as index of disproportionality, in [10] it is obtained the following analytical expression for I_d^* for a specific election

$$I_d^* = \frac{200}{V} \sum_{j=1}^{\Delta M} (Q - R_j) = 200 \left(\frac{\Delta M}{M} - \frac{1}{V} \sum_{j=1}^{\Delta M} R_j \right), \tag{3}$$

where $Q = V/M$ is the standard quota (Hare quota), and R_j, $j = \overline{1, \Delta M}$ are the largest ΔM remainders from the $\Delta V_i = V_i - a_i Q$, $i = \overline{1, n}$, $a_i = \lfloor V_i/Q \rfloor$ and

$$\Delta M = \frac{1}{Q} \sum_{i=1}^{n} \Delta V_i. \tag{4}$$

From (3), one can see that I_d^* depends on difference between ratios $\Delta M/M$ and $(R_1 + R_2 + \ldots + R_{\Delta M})/V$.

Mathematical expectancy $\bar{I}_d^*$ depends both, on the specificity of optimization problem, reflected in solution (3), and on the characteristics of the set of ballots for which it is determined. In [10] four approaches are proposed: 1) direct; 2) simplistic, based on the definition of I_d^*; 3) highly simplified, based on a conventional election, the characteristics of which are equal to certain average characteristics of an infinity of polls; 4) simplified, based on $n - 1$ conventional polls, the characteristics of which are equal to certain average characteristics of an infinity of polls. In the following we will examine the first three approaches.

Direct approach involves the calculation of I_{dk}^* value of I_d^* index for each election k, and then the average $\bar{I}_d^*$ value on all K polls. At $K \to \infty$, $\bar{I}_d^*$ becomes mathematical expectancy. The main drawback of this approach is the difficulty of obtaining the analytical solution. The solution can be obtained only by simulation, some of results being described in s. 3.2.

The analytical solution can be obtained in the other three approaches: simplistic, highly simplified and simplified.

Simplistic approach assumes that the distribution of index I_d^* values is a symmetrical one to the middle of its definition domain $[\breve{I}_d^*, \hat{I}_d^*]$, and the $\bar{I}_d^*$ value may be determined as $\bar{I}_d^* = (\breve{I}_d^* + \hat{I}_d^*)/2 = \hat{I}_d^*/2$. Here $\breve{I}_d^* = 0$ is the lower limit and $\hat{I}_d^*$ – the upper limit of the definition

domain. According to [10], in case of this approach $\bar{I}_d^*$ is calculated as follow

$$\bar{I}_d^* = \frac{\widehat{I}_d^*}{2} = \frac{25}{M} \left\{ \begin{array}{l} n, \text{ at } n \text{ even} \\ n - \frac{1}{n}, \text{ at } n \text{ odd} \end{array} \right. , \quad \% \text{ of seats.} \tag{5}$$

From (5) it results that function $\bar{I}_d^*(M,n)$ is monotonically decreasing to M and monotonically increasing to n, and at even values of n and $M = n$ it does not depend on M and n. The upper limit of $\bar{I}_d^*(M,n)$, taking into account that $n \leq M$, is obtained at $M = n$: at even values of n, it does not depend on $M = n$ and is equal to 25%; at odd values of n, it increases with increasing of $M = n$ (since $sign(\partial \bar{I}_d^*/\partial n) = sign(n) > 0$ at $n \geq 3$) from 200/9% $\approx 22,22\%$, for $M = n = 3$, and tending to 25% for $M = n \to \infty$.

The highly simplified approach involves the use, as average value of I_d^* for an infinite number of ballots K, of the value $\tilde{I}_d^*$ for the conventional election with average remainders $\tilde{R}_j$, $j = \overline{1,n}$. According to [10], in case of this approach $\tilde{I}_d^*$ is determined as follows

$$\tilde{I}_d^* = \frac{25}{M} \left\{ \begin{array}{l} n, \text{ at } n \text{ even} \\ (n+1)(1 - \frac{1}{n^2}), \text{ at } n \text{ odd.} \end{array} \right. \tag{6}$$

From (6) it can be easily seen that function $\tilde{I}_d^*(M,n)$ is monotonically decreasing to M and monotonically increasing to n, and at even values of n and $M = n$ it does not depend on M and n. The upper limit of $\tilde{I}_d^*(M,n)$, taking into account that $n \leq M$, is obtained at $M = n$: at even values of n, it does not depend on $M = n$ and is equal to 25%; at odd values of n, it increases with the decreasing of $M = n$ (since $sign(\partial \tilde{I}_d^*(M,n)/\partial n) = sign(-n^2 + 2n + 3) < 0$ at $n > 3$) from 200/9% $\approx 22,22\%$, for $M = n = \infty$, and tending to 800/27% $\approx 29,63\%$ at $M = n = 3$.

Comparing expressions (5) and (6), it can be seen that for even n they coincide. Moreover, at $n = 2$, because always $\Delta M = 1$, these expressions convey exactly the average I_d^* value, i.e.

$$\tilde{I}_d^*\Big|_{n=2} = \frac{50}{M}\%. \tag{7}$$

3.2 Mathematical expectancy of disproportionality, by simulation

SIMOD application performs the direct approach for determining the disproportionality of seats allocation, according to Hamilton method, by computer simulation. The methodology for multi-optional PR voting systems computer simulation is described in [6]. Subject to simulation is only quantities V_i, $i = \overline{1,n}$.

The simulation was carried out, using the following initial data: N= 200000 (sample); $V = 100000000$; $M = 5, 10, 20, 50, 100$; $n = 2, 3, 4, 5, 7, 10, 15, 20, 50, 100$, $n \leq M$. When generating quantities V_i, $i = \overline{1,n}$, the uniform distribution was used. Some of the results of calculations are shown in Table 1.

Table 1. Average value $\bar{I}^*_{ds}$ of I^*_d, obtained by simulation, %

Seats,	Number of parties, n									
M	2	3	4	5	7	10	15	20	50	100
5	10,148	15,526	21,250	26,347						
10	4,980	7,790	10,419	13,042	18,026	26,350				
20	2,498	3,890	5,210	6,505	9,056	12,827	19,021	25,984		
50	0,998	1,555	2,086	2,600	3,620	5,135	7,646	10,148	25,504	
100	0,500	0,778	1,042	1,300	1,811	2,569	3,823	5,073	12,651	25,277

From Table 1 it can be seen that the $\bar{I}^*_{ds}$ value is decreasing both, to the number of seats M and to the number of parties n, the maximum value (about 25-26%) being reached at $M = n$. In practice, as a rule, cases for which $M = n$ are not met. Ratio n/M does not exceed, usually, 0,1 and then, as it is shown in Table 1, $\bar{I}^*_{ds} \leq 3\%$. Complementary to data of Table 1, for $M = 200$ and $n = 20$, it was obtained the value $\bar{I}^*_{ds} = 2,537\%$, down from $\bar{I}^*_{ds} = 5,073\%$ for $M = 100$ and $n = 20$. Data from Table 1 at $n = 2$ also confirm justice of estimate (7).

3.3 Comparative analyses

For comparative analysis of the three approaches: direct, simplistic and highly simplified, the essence of which is described in s. 3.1, in sections (a) and (b) of Table 2 there are shown respective quantitative values for the same sets of initial data values as those used for Table 1 in s. 3.2.

Table 2. Average value of I_d^* index conform to (5), (6) and combined

Seats,	Number of parties, n									
M	2	3	4	5	7	10	15	20	50	100
	a) Simplistic approach ($\bar{I}_d^*$), %									
5	10	13,333	20	24						
10	5	6,667	10	12	17,143	25				
20	2,5	3,333	5	6	8,571	12,5	18,667	25		
50	1	1,333	2	2,4	3,429	5	7,467	10	25	
100	0,5	0,667	1	1,2	1,714	2,5	3,733	5	12,5	25
	b) Highly simplified approach ($\tilde{I}_d^*$), %									
5	10	17,778	20	28,8						
10	5	8,889	10	14,4	19,592	25				
20	2,5	4,444	5	7,2	9,796	12,5	19,911	25		
50	1	1,778	2	2,88	3,918	5	7,964	10	25	
100	0,5	0,889	1	1,44	1,959	2,5	3,982	5	12,5	25
	c) Combined approach ($\bar{I}_{dc}^*$), %									
5	10	15,555	21,094	26,4						
10	5	7,778	10,547	13,200	18,367	25,988				
20	2,5	3,889	5,273	6,600	9,184	12,994	19,289	25,561		
50	1	1,556	2,110	2,640	3,674	5,198	7,716	10,224	25,240	
100	0,5	0,778	1,055	1,320	1,837	2,599	3,858	5,112	12,620	25,123

Comparing data of sections (a) and (b) in Table 2 with those of Table 1, it can be seen that, for odd values of n, there are the following relations: $\bar{I}_d^* < \bar{I}_{ds}^* < \tilde{I}_d^*$, and for the even ones, except $n = 2$, on the contrary, $\bar{I}_{ds}^* > \bar{I}_d^* = \tilde{I}_d^*$. Therefore, section (c) of Table 2 presents data for the combined approach ($\bar{I}_{dc}^*$), the average ratio, in this case, being calculated according to formula

$$\bar{I}^*_{dc} = \frac{25}{M} \begin{cases} n, \text{ at } n = 2 \\ (n + \frac{1}{2})(1 - \frac{1}{n^2}), \text{ at } n > 2, \end{cases} \tag{8}$$

where $(n+1/2)(1-1/n^2) = [(n-1/n)+(n+1)(1-1/n^2)]/2$ from the second line of (5) and (6).

To compare the direct approach (by simulation) with theoretical approaches for each variant of the initial data, the absolute value of difference between the value of each element of Table 1 with that of the respective element of Table 2 is calculated. The obtained results are shown in Table 3.

Table 3. Value of difference between $\bar{I}^*_{ds}$ and indices values from Table 2

Seats,	Number of parties, n									
M	2	3	4	5	7	10	15	20	50	100
	a) Absolute value of difference $\bar{I}^*_{ds} - \bar{I}^*_{d}$, %									
5	0,148	2,192	1,250	2,347						
10	0,020	1,123	0,419	1,0420	0,883	1,351				
20	0,002	0,557	0,210	0,505	0,484	0,327	0,355	0,984		
50	0,002	0,222	0,086	0,200	0,191	0,135	0,179	0,148	0,504	
100	0,001	0,111	0,042	0,100	0,096	0,069	0,090	0,073	0,151	0,277
	b) Absolute value of difference $\bar{I}^*_{ds} - \tilde{I}^*_{d}$, %									
5	0,148	2,252	1,250	2,453						
10	0,020	1,099	0,419	1,358	1,566	1,351				
20	0,002	0,554	0,210	0,696	0,740	0,327	0,890	0,984		
50	0,002	0,223	0,086	0,280	0,299	0,135	0,319	0,148	0,504	
100	0,001	0,111	0,042	0,140	0,149	0,069	0,159	0,073	0,151	0,277
	c) Absolute value of difference $\bar{I}^*_{ds} - \bar{I}^*_{dc}$, %									
5	0,148	0,030	0,156	0,053						
10	0,020	0,012	0,128	0,158	0,341	0,363				
20	0,002	0,001	0,064	0,095	0,128	0,167	0,268	0,423		
50	0,002	0,001	0,024	0,040	0,054	0,063	0,070	0,076	0,264	
100	0,001	0,000	0,012	0,020	0,026	0,030	0,035	0,039	0,031	0,154

Comparing data of sections (a)-(c) of Table 3, it can be seen that the lowest absolute deviations from results, obtained by simulation, are for combined approach ($\bar{I}^*_{ds} - \bar{I}^*_{dc}$). Such deviations do not exceed, for

$M = n$, approx. 0,5%. At the same time, for cases encountered in practice (as a rule, $n/M \leq 0,1$), as shown in section (c) of Table 3, $|\bar{I}^*_{ds} - \bar{I}^*_{dc}| \leq 0,04\%$.

So, to forecasting the average disproportionality of seats allocation ($\bar{I}^*_d$), when applying the Hamilton method, it is appropriate to use the expression (8), the error not exceeding 0,5% of seats, and in most practical cases – 0,05% of seats.

4 Comparison of monotone VD methods by simulation

There are compared the well known d'Hondt, Sainte-Laguë and Huntington-Hill monotone VD methods. Some results of calculations at uniform distribution of quantities V_i, $i = \overline{1,n}$ and initial data: $M = 5, 10, 20, 50, 100$; $n = 3, 4, 5, 7, 10, 15, 20, 50$ $(n < M)$; $V = 10^8$; sample size of 200000 ballots for each pair $\{M, n\}$, are presented in Figs. 1–3.

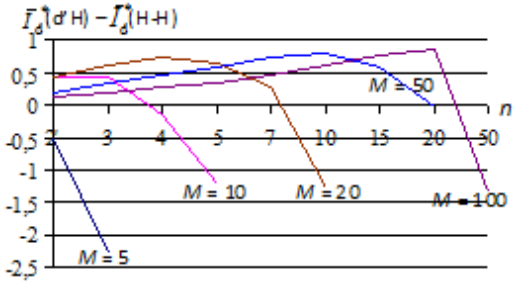


Figure 1. $\bar{I}^*_d$(dHondt) $-\bar{I}^*_d$(Huntington-Hill), %.

From Fig. 1 one can see that, by parameter $\bar{I}^*_d$, in some cases the Huntington-Hill method is better then the d'Hondt one and, in other cases, vice versa. Also, with the increase of M and decrease of n the Huntington-Hill method became better than the d'Hondt one. At the same time, for cases encountered in practice (usually, $n/M \leq 0,1$),

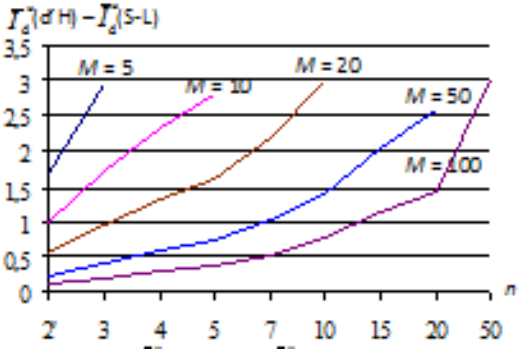


Figure 2. $\bar{I}_d^*$ (dHondt) $-\bar{I}_d^*$(Sainte-Lague), %.

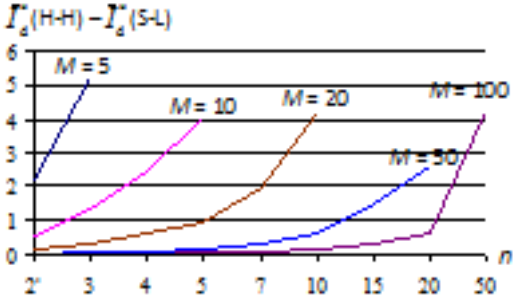


Figure 3. $\bar{I}_d^*$(Huntington-Hill) $-\bar{I}_d^*$(Sainte-Lague), %.

excepting $n = 2$ for small values of M, Huntington-Hill method is better than the d'Hondt one ($\bar{I}_d^*(d'H) - \bar{I}_d^*(H - H) > 0$).

Similarly, from Figures 2 and 3 one can see that Sainte-Laguë method is better then the d'Hondt and Huntington-Hill ones, no matter of parameters M and n values ($\bar{I}_d^*(d'H) - \bar{I}_d^*(S - L) > 0$, $\bar{I}_d^*(H - H) - \bar{I}_d^*(S - L) > 0$). Also, the value of differences $\bar{I}_d^*(d'H) - \bar{I}_d^*(S - L)$ and $\bar{I}_d^*(H - H) - \bar{I}_d^*(S - L)$ are increasing with the decrease of M and the increase of n.

So, the best, by parameter $\bar{I}_d^*$, from the examined monotone methods, is the Sainte-Laguë one. However, there may be particular cases, when the Huntington-Hill method, as well as the d'Hondt method, ensures a lower value of parameter I_d^* then the Sainte-Laguë one. To characterize such situations, parameters R_{SL-dH} and R_{SL-HH} are used. Parameter R_{SL-dH} is the ratio of the percentage of ballots, for which $\bar{I}_d^*$ (d'Hondt) $> \bar{I}_d^*$(Sainte-Laguë), to the percentage of ballots, for which $\bar{I}_d^*$(d'Hondt) $< \bar{I}_d^*$ (Sainte-Laguë). Similarly, R_{SL-HH} is the ratio of the percentage of ballots, for which $\bar{I}_d^*$(Huntington-Hill) $> \bar{I}_d^*$(Sainte-Laguë), to the percentage of ballots, for which $\bar{I}_d^*$(Huntington-Hill) $< \bar{I}_d^*$(Sainte-Laguë).

Some results of parameters R_{SL-dH} and R_{SL-HH} calculations, at uniform distribution of quantities V_i, $i = \overline{1,n}$ and initial data: $M = 20, 100$; $n = 3, 4, 5, 10$; $V = 10^8$; sample size of 200000 ballots for each pair $\{M, n\}$, are systemized in Table 1. Here P is the percentage of ballots, for which $\bar{I}_d^*$(Sainte-Laguë) = $\bar{I}_d^*$(Hamilton)

Table 4. Some results of parameters P, R_{SL-dH} and R_{SL-HH} calculation

	$M = 20$				$M = 100$			
	n =3	n =4	n =5	n =10	n =3	n =4	n =5	n =10
P, %	94,04	91,28	89,03	85,63	94,02	91,43	89,09	81,60
R_{SL-dH}, times	12,84	13,46	15,57	32,11	12,82	14,41	16,59	37,60
R_{SL-HH}, times	4,79	5,71	6,84	25,49	2,68	3,19	3,65	5,38

From Table 1 one can see that, in cases of examined initial

data, for more than 80% of ballots Sainte-Laguë method gives the same allocation of seats as the Hamilton one does ($\bar{I}_d^*$(Sainte-Laguë) = $\bar{I}_d^*$(Hamilton)). Also, Sainte-Laguë method gives a better distribution of seats for a number of polls at least 12-38 times higher than the d'Hondt one and at least of 2,525 times higher than the Huntington-Hill one does. Elsewhere, P index is decreasing and R index is increasing with the increase of the number n of parties: more parties – less efficient is Sainte-Laguë method in comparison with the d'Hondt and Huntington-Hill ones.

More than that, the Sainte-Laguë method meets the lower quota ($x_i \geq a_i$, $i = \overline{1,n}$), while the Huntington-Hill one may not satisfy it [3]. Another advantage of Sainte-Laguë method, at $n = 2$ it coincides with the Hamilton (optimal) one, while the d'Hondt and Huntington-Hill may not coincide [5]. Also, at $n = 2$ and $n = 3$, the Sainte-Laguë method meets the quota rule ($a_i \leq x_i \leq a_i + 1$, $i = \overline{1,n}$), while the d'Hondt and Huntington-Hill ones may not satisfy it [7]

Additionally, as it is shown in [11], Webster method is not always equivalent, as affirmed in [1] and other publications, to the Sainte-Laguë one. For equivalence, Webster method needs, in some cases, additional operations shown in [11]. As a result, the use of Sainte-Laguë method is easier than that of such complemented Webster method.

Overall, the use of Sainte-Laguë method is more efficient than that of d'Hondt and Huntington-Hill ones and is easier than that of complemented Webster method.

5 The adapted Sainte-Laguë method

In some cases it is needed to allocate to each party a number of seats not lower than an established value. For example, in the United States Congress House of Representatives each state shall have at least one representative (seat). The ordinary Sainte-Laguë method does not ensure such allocation of seats. But it is ensured by Huntington-Hill method and the described in this section adapted Sainte-Laguë (ASL) method.

The adapted Sainte-Laguë method differs from the Sainte-Laguë

one only by satisfying the condition that each state shall have at least one representative (seat) in the House. According to this method, the allocation of seats to states shall be done as follows:

1. Let Ω be the set of states, $|\Omega| = n$. Allocating of seats to states with the number of population that do not exceed the standard quota (Hare quota) $Q = V/M$: for $i = \overline{1,n}$, if $V_i \leq Q$, then $x_i := 1$, $V := V - V_i$, $M := M - 1$, $\Omega := \Omega/i$

2. For the new set Ω of states and new values of parameters $n = |\Omega|$, M, V and $Q = V/M$, to allocate seats according to the ordinary Sainte-Laguë method. Stop.

Statement 5.1. The adapted Sainte-Laguë method is immune to the Alabama, of Population and of New State paradoxes.

Proof. With refer to Alabama paradox, we have: $M' > M$; $V_i' = V_i$, $i = \overline{1,n}$. Therefore, $V' = V$, $V'/M' = Q' < Q = V/M$. Here by stroke (') the parameters for the second ballot are noted. To avoid the Alabama paradox, it shall be $x_i' \geq x_i$, $i = \overline{1,n}$. Indeed, let $F_j = V_j/[2(x_j - 1) + 1] = \min\{V_i/[2(x_i - 1) + 1], i \in \Omega/G\}$, where G is the set of parties for which $V_i \leq$ and $x_i = 1$. Because of $Q' < Q$ and $V_i' = V_i$, $i = \overline{1,n}$, the number of parties for which $V_i' \leq Q'$ is not larger than that of set G. So, for states of set G, having $x_i = 1$, the condition $x_i' \geq x_i$ is satisfied. From the other hand, because of $M' > M$, it takes place $F_k' \leq F_j$, where $F_k' = \min\{V_i'/[2(x_i' - 1) + 1]\}$, $i \in \Omega/G$. That is why for states of set Ω/G the condition $x_i' \geq x_i$ is satisfied, too.

For the case of Population paradox, we have: $M' = M$; $V_i' = V_i$, $i = \overline{1,n}/k$; $V_k' > V_k$. Therefore, $V' = V + V_k' - V_k$, $V'/M' = Q' > Q = V/M$. To avoid the Population paradox, it shall be $x_k' \geq x_k$. Indeed, if $V_k \leq Q$, then $x_k = 1$ and because of restriction $(x_k, x_k') \geq 1$ anyway occurs $x_k' \geq x_k$. Let $V_k > Q$, then $V_k' > Q'$, too, because of $Q' = (V + V_k' - V_k)/M$ and $V_k' = V_k + (V_k' - V_k)$, from where $V_k' - Q' = V_k - Q + (V_k' - V_k)(1 - 1/M) > 0$, taking into account that $V_k' > V_k$ and $M > 1$. If so, and taking into account that $Q' > Q$, the number of parties for which $V_i' > Q$' is not larger than that of set Ω/G. Therefore, and taking into account that $V_k' > V_k$, we have $F_k' > F_j$, where $F_k' = V_k'/[2(x_k' - 1) + 1]$ and $F_j = V_j/[2(x_j - 1) + 1] = \min\{V_i/[2(x_i - 1) + 1], i \in \Omega/G\}$ So, in case of $V_k > Q$ the condition

$x'_k \geq x_k$ is satisfied, too.

Finally, regarding the New State paradox, we have: $V'_i = V_i$, $i = \overline{1,n}$; $V_{n+1} = 0$; $V'_{n+1} > 0$; $V' = V + V'_{n+1}$; $M' = M + x_{n+1}$. The value of x_{n+1} is obtained by applying the respective optimization method. To avoid the New State paradox, it shall be $x'_i = x_i$, $i = \overline{1,n}$. Indeed, depending on relation between Q and Q' it may be, for the second ballot, that some states from set G will move to set Ω'/G' (if $Q > Q'$) or on the contrary some states from set Ω/G will move to set G' (if $Q < Q'$), or $G' = G \cup (n+1)$ (if $Q = Q'$) and no states move between these sets. But in no cases these movements of states between mentioned above sets do not influence the functions of preference of parties $F_i = V_i/[2(x_i - 1) + 1]$, $i = \overline{1,n}$ and therefore they do not change the allocation of seats to parties.

It is easy to observe that the adapted Sainte-Laguë method is immune to the Alabama, of Population and of New State paradoxes also in cases when it is needed to allocate to each party a number of seats not lower than an arbitrary nonnegative value, including larger than one.

6 A case study

From 1941, the allocation of seats for the United States Congress House of Representatives (the apportionment) is done using the Huntington-Hill method. Let us compare, the Huntington-Hill and the described in section 5 adapted Sainte-Laguë methods, when applied for apportionment.

In Table 5 there are systemized data of apportionment for US Census population in the period of 1940-2010 years and year 2014, being used the following notations:

ΔX_{HH} – the number of seats by which the Huntington-Hill apportionment differs from the optimal Hamilton one;

ΔX_{ASL} – the number of seats by which the adapted Sainte-Laguë apportionment differs from the optimal Hamilton one.

From Table 5 one can see that only in one from nine cases of apportionment, the adapted Sainte-Laguë method gets a less proportional

Table 5. Data of apportionment by Huntington-Hill and ASL methods

	ΔX_{HH}	ΔX_{ASL}	$\Delta X_{HH} - \Delta X_{ASL}$	States for which x_i differs, and its deviation from the optimal value, obtained by Hamilton method	
				By Huntington-Hill method	By adapted Sainte-Laguë method
1940	2	2	0	Arkansas: -1, Nevada: +1	Arkansas: -1, Nevada: +1
1950	6	4	2	California: -1, Massachusetts: -1, Arkansas: -1, Hawaii: +1, Nevada: +1, Alaska: +1	California: -1, Arkansas: -1, Nevada: +1, Alaska: +1
1960	4	0	4	Illinois: -1, Massachusetts: -1, West Virginia: +1, New Hampshire: +1	**None**
1970	4	0	4	Illinois: -1, North Carolina: -1, Idaho: +1, Montana: +1	**None**
1980	4	2	2	California: -1, Massachusetts: -1, New Mexico: +1, Montana: + 1	California: -1, Montana: +1
1990	4	2	2	New York: -1, New Jersey: -1, Oklahoma: +1, Mississippi: +1	New York: -1, Washington: +1
2000	2	0	2	North Carolina: -1, Utah: +1	**None**
2010		2	-2	**None**	North Carolina: +1, Rhode Island: -1
2014	2	0	2	Pennsylvania: -1, Rhode Island: +1	**None**

result (two seats) than the Huntington-Hill method does (year 2010), when the Huntington-Hill method gets a less proportional result (two or four seats) then the adapted Sainte-Laguë method does in seven cases (years 1950, 1960, 1970, 1980, 1990, 2000 and 2014). In four cases the apportionments obtained by adapted Sainte-Laguë method coincide with the optimal Hamilton (years 1960, 1970, 2000 and 2014), when the apportionment obtained by adapted Huntington-Hill method coincide with the optimal Hamilton only in one case (year 2010). In one case (year1940), the apportionments, obtained by both compared methods, coincide. So, these particular cases confirm the fact that the adapted Sainte-Laguë method is considerably better than the Huntington-Hill one.

Let's look beyond. We will compare the Huntington-Hill and the adapted Sainte-Laguë methods, by computer simulation using the SIMOD application, for the same US Census years and the same US summary states population (V), but for general uniform or standard normal distribution of states population V_i, $i = \overline{1,n}$. As comparison criterion we will use R_{ASL-HH} – the ratio of the percentage of ballots, for which $\bar{I}_d^*$ (adapted Sainte-Laguë) $< \bar{I}_d^*$ (Huntington-Hill), to the percentage of ballots, for which $\bar{I}_d^*$ (Huntington-Hill) $< \bar{I}_d^*$ (adapted Sainte-Laguë)

The computer simulation was carried out using samples of 200000 ballots each. So, for uniform distribution of states population we have R_{ASL-HH} $(1940) = 6,720$ times and for the other eight years – $R_{ASL-HH} \in [7,276; 7,535]$ times. The value for the year 1940 differs essentially from values for the other eight years because in that year there were only 48 states. For standard normal distribution of states population, we have $R_{ASL-HH}(1940) = 2,183$ times and for the other eight years – $R_{ASL-HH} \in [2,258; 2,306]$ times. This data also confirm that, for apportionment, the adapted Sainte-Laguë method is considerably better than the Huntington-Hill one.

It was also determined that, at uniform distribution of states population, apportionments with adapted Sainte-Laguë method have the average value of I_d^* index $\bar{I}_d^*(ASL, 1940) = 2,946$ % of seats and for the other eight years – $\bar{I}_d^*(ASL) \in [3,074; 3,076]$ % of seats. Sim-

ilar, at standard normal distribution of states population, we have $\bar{I}_d^*(ASL, 1940) = 2,785$ % of seats and for the other seven Census years and year 2014 – $\bar{I}_d^*(ASL) \in [2,899; 2,900]$ % of seats.

7 Conclusions

Combining theoretical results and computer simulation, the formula is obtained for the estimation of average disproportion of seats allocation, using Hamilton method, for each particular value of the number M of seats and of the number n of parties, the error not exceeding 0,5% of seats, and in most practical cases – 0,05% of seats. This formula can be used to predict the disproportionality of multi-optional decisions by Hamilton method.

By proportionality of voters' will representation in the final multi-optional decision, from the three compared monotone VD methods with divisor – d'Hondt, Huntington-Hill and Sainte-Laguë, the best is the last one. For example, in cases of examined initial data ($20 \leq M \leq 100, 3 \leq n \leq 10$ and uniform distribution of quantities V_i, $i = \overline{1,n}$) the Sainte-Laguë method gives a better distribution of seats for a number of polls at least 12-38 times higher than the d'Hondt one and at least of 2,5-25 times higher than the Huntington-Hill one does. Also, the use of Sainte-Laguë method is easier than that of complemented Webster method.

The Sainte-Laguë method is adapted to the requirement that each party (state) shall have at least one seat (representative) in the elective body. It is proved that the adapted Sainte-Laguë method is immune to the Alabama, of Population and of New State paradoxes. By computer simulation it is shown that ASL method is considerably better, in sense of minimizing the disproportion of seats allocation, than the Huntington-Hill one. So, for the US Census summary states population (V) and uniform distribution of states population $V_i i = \overline{1,n}$, the ASL method gives, in average, a better allocation of seats for a number of polls of 6,720 times higher for the Census year 1940, and of 7,276-7,535 times higher than the Huntington-Hill one does, for the other seven Census years in the period of 1950-2010 years (1950, 1960, ...,

2010) and year 2014. In case of standard normal distribution of states population V_i, $i = \overline{1,n}$, the ASL method gives, in average, a better allocation of seats for a number of polls of 2,183 times higher for the Census year 1940, and of 2,258-2,306 times higher than the Huntington-Hill one does, for the other seven Census years in the period of 1950 – 2010 years (1950, 1960, ..., 2010) and year 2014. Apportionments with adapted Sainte-Laguë method have the average value of I_d^* index equal to approximately $2,9 - 3,0$ seats.

These average results were confirmed by conventional apportionment for the United States Congress House of Representatives for Census values of states population V_i, $i = \overline{1,n}$ in the examined period: only in one, from the nine cases of apportionment, the ASL method gets a less proportional result than Huntington-Hill method does (year 2010), while the Hunting-ton-Hill method gets a less proportional result than the ASL method does in seven cases (years 1950, 1960, 1970, 1980, 1990, 2000 and 2014).

So, from the point of view of US Constitution requirement of proportional representation of states in the United States Congress House of Representatives, it is considerably better to use for apportionment the adapted Sainte-Laguë method than the used from 1941 year Huntington-Hill method.

References

[1] M. Gallagher, P. Mitchell. *The Politics of Electoral Systems*. London: Oxford University Press, 2008.

[2] I. Bolun. *Comparison of indices of disproportionality in PR systems*. Computer Science Journal of Moldova, vol.20, no 2, 2012. – pp. 246–271.

[3] I. Bolun. *"Votes-decision" monotone method in PR systems*. Economica, no 4(78), 2011. – pp. 108–117.

[4] I. Bolun. *Algorithmization of optimal allocation of seats in PR systems*. Economica, nr.3(77)/2011. – pp. 137–152.

[5] I. Bolun. *Disproportionality of some"votes-decision" rules in PR systems*. In: 20 years of economical reforms, intern. conf., Sept. 23-24, 2011. Vol. I. Chisinau: Editura ASEM, 2011. – pp. 425–430 (Romanian).

[6] I. Bolun, S. Cebotari. *An application for PR voting systems simulation*. In: 60 years of economical higher education in the Republic of Moldova, intern. conf., Sept. 27-28, 2013. Vol. I. Chisinau: Editura ASEM, 2013. – pp. 372–376 (Romanian).

[7] I. Bolun. *Disproportionality of multi-optional PR voting systems*. In: The Third Conference of Mathematical Society of the Republic of Moldova, Aug. 19-23, 2014, Chisinau, Moldova: Proceedings IMCS-50. Chisinau: IMI, 2014. – pp. 471–476.

[8] I. Bolun. *Seats allocation in party-list elections*. Economica, nr.2(76)/2011. – pp. 138–151.

[9] I. Bolun. *Comparison of indices of disproportionality in PR systems*. Computer Science Journal of Moldova, vol. 20, no. 2(59), 2012. – pp. 246–271.

[10] I. Bolun. *Definition domain of optimal solution disproportionality in PR systems*. In: Annals of Academy of Economic Studies of Moldova, Ed. 10. Chisinau: Editura ASEM, 2012. – pp. 283–298. (Romanian).

[11] I. Bolun. *About equivalence of Jefferson and d'Hondt methods and, respectively, of that of Webster and Sainte-Laguë ones*. In: Competitiveness and innovation in the knowledge economy, intern. conf., Sept. 28-29, 2012. Vol. II. Chisinau: Editura ASEM, 2012. – pp. 10–13. (Romanian).

Ion Bolun, Alexandru Costas,

Academy of Economic Studies of Moldova
E–mail: bolun@ase.md

Received July 12, 2015

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Axiomatics for multivalued dependencies in table databases: correctness and completeness

Dmitriy Bui, Anna Puzikova

Abstract

Axiomatics for multivalued dependencies in table databases and axiomatics for functional and multivalued dependencies are reviewed. For each axiomatic relations of syntactic and semantic succession are considered. A rigorous and convincing proof of correctness and completeness of these axiomatics (within the paradigm of mathematical logic) is established. In particular, the properties of closures of sets of specified dependencies are investigated. The properties of set-theoretic function restriction have been used as mathematical framework.

Keywords: table databases, functional dependencies, multivalued dependencies, completeness of axiomatic system.

1 Introduction

In spite of the accumulated theoretical researches normalization theory is fragmented and is far from satisfactory conclusion. The works devoted to the ways of solving existing problems of designing database schemas (see, for example, [1]) and improvement of algorithmic systems for normalization (see, for example, [2]) evidence this fact. The process of normalization is based upon functional and multivalued dependencies theory the foundation of which is made by corresponding axiomatics and their completeness. The overview of research sources has shown that these axiomatics lack the proof of completeness that will comply with mathematical rigor.

2 Axiomatic for Multivalued Dependencies

All undefined concepts and notations are used in understanding of monograph [3], in particular, $s|X$ — restriction of the row s to the set of attributes X.

Let t — a table, R — the scheme of the table t (finite set of attributes); X, Y, W, Z — subsets of scheme R; s, s_1, s_2 — the rows of table t. Henceforth we shall assume that set R and *universal domain* D (the set, from which attributes take on values in interpretations) are fixed.

A multivalued dependence (MVD) $X \to\to Y$ is valid on the table t of the scheme R (see, for example, [3]), if for two arbitrary rows s_1, s_2 of table t which coincide on the set of attributes X, there exists row $s_3 \in t$ which is equal to the union of restrictions of the rows s_1, s_2, to the sets of attributes $X \cup Y$ and $R \setminus (X \cup Y)$ respectively:

$$(X \to\to Y)(t) = true \overset{def}{\Leftrightarrow} \forall s_1, s_2 \in t(s_1|X = s_2|X \Rightarrow \exists s_3 \in t(s_3 = \\ = s_1|(X \cup Y) \cup s_2|R \setminus (X \cup Y))).$$

Structure of table t, which complies with MVD $X \to\to Y$, can be represented using the following relation. We say that rows s_1, s_2 of table t are in the relation $=_X$, if they coincide on the set of attributes X:

$$s_1 =_X s_2 \overset{def}{\Leftrightarrow} s_1|X = s_2|X.$$

It is obvious that relation $=_X$ is equivalence relation and therefore it partitions the table t into equivalence classes, which are as follows:

$$[s]_{=_X} = \{s|X\} \bigotimes \pi_Y([s]_{=_X}) \bigotimes \pi_{R\setminus(X\cup Y)}([s]_{=_X}),$$

where s — arbitrary representative of the class.

A table $t(R)$ is the model of a set of MVD's G, if each MVD $X \to\to Y \in G$ is valid on table $t(R)$:

$$t(R) \textit{ is the model of } G \overset{def}{\Leftrightarrow} \\ \overset{def}{\Leftrightarrow} \forall (X \to\to Y)(X \to\to Y \in G \Rightarrow (X \to\to Y)(t) = true).$$

The following axioms and inference rules are valid [4].

Axiom of *reflexivity*: $\forall t(X \to\to Y)(t) = true$, where $Y \subseteq X$.

Axiom: $\forall t(X \to\to Y)(t) = true$, where $X \cup Y = R$.

Rule of *complementation*: $(X \to\to Y)(t) = true \Rightarrow (X \to\to R \setminus (X \cup$

$Y))(t) = true$.
Rule of *augmentation*: $(X \to\to Y)(t) = true \& Z \subseteq W \Rightarrow (X \cup W \to\to Y \cup Z)(t) = true$.
Rule of *transitivity*: $(X \to\to Y)(t) = true \,\&\, (Y \to\to Z)(t) = true \Rightarrow (X \to\to Z \setminus Y)(t) = true$.

The proof in terms of the monograph [3], for example, the axiom of reflexivity is given in [5].

A MVD $X \to\to Y$ is *semantically* deduced from the set of MVD's G, if at each table $t(R)$, which is the model of set G, MVD $X \to\to Y$ is valid too:

$$G \models X \to\to Y \overset{def}{\Leftrightarrow} \forall t(R)(t \text{ is the model of the } G \Rightarrow (X \to\to Y)(t) = true).$$

The relation $\models$ will be called semantic consequence relation.

From above-mentioned axioms and inference rules follow corollaries.

Lemma 1. *The following properties of the semantic consequence relation are valid:*

1. $\emptyset \models X \to\to Y$ *for* $Y \subseteq X$.
2. $\emptyset \models X \to\to Y$ *for* $X \cup Y = R$.
3. $G \models X \to\to Y \Rightarrow G \models X \to\to R \setminus (X \cup Y)$.
4. $G \models X \to\to Y \,\&\, Z \subseteq W \Rightarrow G \models X \cup W \to\to Y \cup Z$.
5. $G \models X \to\to Y \,\&\, G \models Y \to\to Z \Rightarrow G \models X \to\to Z \setminus Y$.
6. $G \models X \to\to Y \,\&\, G \models Y \to\to Z \,\&\, Z \cap Y = \emptyset \Rightarrow G \models X \to\to Z$.

A MVD $X \to\to Y$ is *syntactically* derived from the set of MVD's G with respect to the scheme R ($G \vdash_R X \to\to Y$), if there is a finite sequence of MVD's $\varphi_1, \varphi_2, \ldots, \varphi_{m-1}, \varphi_m$, where $\varphi_m = X \to\to Y$ and for all $\forall i = \overline{1, m-1}$ each φ_i is either the axiom of reflexive or belongs to G, or is derived with some inference rule for MVD's (complementation, augmentation, transitivity) from the previous in this sequence φ_j, φ_k, $j, k < i$.

Let sequence $\varphi_1, \varphi_2, \ldots, \varphi_{m-1}, \varphi_m$ be called proof, following the tradition of mathematical logic [6].

Let there be given certain set of MVD's G. *Closure* $[G]_R$ is a set of all MVD's, that are syntactically derived from G:

$$[G]_R \overset{def}{=} \{X \to\to Y | G \vdash_R X \to\to Y\}.$$

For notational convenience, we write $\vdash$ for $\vdash_R$.

Lemma 2. *The following properties are valid:*
1) $G \subseteq [G]$ *(increase);*
2) $[[G]] = [G]$ *(idempotency);*
3) $G \subseteq H \Rightarrow [G] \subseteq [H]$ *(monotonicity).*

The proofs of this properties are given in [5].

Thereby, operator $G \mapsto [G]$ is closure operator in terms of [7].

From reflexivity axiom and inference rules indicated above it is possible to get other inference rules for MVD's [4].

Rule of *pseudo-transitivity*:

$\{X \to\to Y, Y \cup W \to\to Z\} \vdash X \cup W \to\to Z \setminus (Y \cup W)$.

Rules of *difference*:

1. $\{X \to\to Y\} \vdash X \to\to Y \setminus X$;
2. $\{X \to\to Y \setminus X\} \vdash X \to\to Y$;
3. $\{X \to\to Y\} \vdash X \to\to R \setminus Y$.

Rule of *union*: $\{X \to\to Y_1, X \to\to Y_2\} \vdash X \to\to Y_1 \cup Y_2$.

Rules of *decomposition*:

1. $\{X \to\to Y_1, X \to\to Y_2\} \vdash X \to\to Y_1 \cap Y_2$;
2. $\{X \to\to Y_1, X \to\to Y_2\} \vdash X \to\to Y_1 \setminus Y_2$.

Lemma 3. *The following properties are valid for* $n = 2, 3, \ldots$*:*
1. $\{X \to\to Y_1, \ldots, X \to\to Y_n\} \vdash X \to\to Y_1 \cup \ldots \cup Y_n$*;*
2. $\{X \to\to Y_1, \ldots, X \to\to Y_n\} \vdash X \to\to Y_1 \cap \ldots \cap Y_n$.

The proof of this lemma is constructed by the induction in the n, according to the rules of augmentation and transitivity.

3 Axiomatic for FD's and MFD's

It will be recalled that a functional dependence $X \to Y$ is valid on the table t, if for two arbitrary rows s_1, s_2 of table t which coincide on the set of attributes X, their equality on the set of attributes Y is fulfilled (see, for example [3]), that is:

$$(X \to Y)(t) = true \overset{def}{\Leftrightarrow} \forall s_1, s_2 \in t(s_1|X = s_2|X \Rightarrow s_1|Y = s_2|Y).$$

Let there be given sets F and G of FD's and MVD's respectively. A table $t(R)$ is the *model* of a set $F \cup G$, if each dependency $\varphi \in F \cup G$ is valid on table t:

$$t(R) \text{ is model of } F \cup G \overset{def}{\Leftrightarrow} \forall \varphi(\varphi \in F \cup G \Rightarrow \varphi(t) = true).$$

Mixed inference rules for FD's and MVD's are valid [4].

1. Rule of *extension* FD to MVD: $(X \to Y)(t) = true \Rightarrow (X \to\to Y)(t) = true$.

2. $(X \to\to Z)(t) = true \,\&\, (Y \to Z')(t) = true \,\&\, Z' \subseteq Z \,\&\, Y \cap Z = \emptyset \Rightarrow (X \to Z')(t) = true$.

The proof of extension rule is given, for example, in the monograph [3, p. 73] but the proof of rule 2 — in [5].

FD or MVD φ is *semantically* deduced from the set of dependencies $F \cup G$, if at each table $t(R)$, which is the model of a set of dependencies $F \cup G$, dependency φ is valid too:

$$F \cup G \models \varphi \overset{def}{\Leftrightarrow} \forall t(R)(t \text{ model of } F \cup G \Rightarrow \varphi(t) = true).$$

From above-mentioned mixed inference rules for FD's and MVD's follow corollaries (the properties of semantic consequence relation):

1. $F \models X \to Y \Rightarrow F \models X \to\to Y$;

2. $G \models X \to\to Z \,\&\, F \models Y \to Z' \,\&\, Z' \subseteq Z \,\&\, Y \cap Z = \emptyset \Rightarrow F \cup G \models X \to Z'$.

Lemma 4. *Let H_1 and H_2 — the sets of dependencies (FD's or MVD's) and T_1, T_2 — the sets of all their models respectively. Then implication $H_1 \subseteq H_2 \Rightarrow T_1 \supseteq T_2$ is carried out.*

Corollary 1. *The following properties of the semantic consequence relation are valid:*

1. $F \models \varphi \Rightarrow F \cup G \models \varphi$;
2. $G \models \varphi \Rightarrow F \cup G \models \varphi$.

Lemma 5. *The following properties of the semantic consequence relation are valid:*

1) $F \models X \to Y \Rightarrow F \cup G \models X \cup Z \to Y \cup Z$ *for* $Z \subseteq R$;
$F \cup G \models X \to Y \Rightarrow F \cup G \models X \cup Z \to Y \cup Z$ *for* $Z \subseteq R$;

2) $F \models X \to Y \& F \models Y \to Z \Rightarrow F \cup G \models X \to Z$;
$F \cup G \models X \to Y \& F \cup G \models Y \to Z \Rightarrow F \cup G \models X \to Z$;

3) $G \models X \to\to Y \Rightarrow F \cup G \models X \to\to R \setminus (X \cup Y)$;
$F \cup G \models X \to\to Y \Rightarrow F \cup G \models X \to\to R \setminus (X \cup Y)$;
4) $G \models X \to\to Y \& Z \subseteq W \Rightarrow F \cup G \models X \cup W \to\to Y \cup Z$;
$F \cup G \models X \to\to Y \& Z \subseteq W \Rightarrow F \cup G \models X \cup W \to\to Y \cup Z$;
5) $G \models X \to\to Y \& G \models Y \to\to Z \Rightarrow F \cup G \models X \to\to Z \setminus Y$;
$F \cup G \models X \to\to Y \& F \cup G \models Y \to\to Z \Rightarrow F \cup G \models X \to\to Z \setminus Y$;
6) $F \models X \to Y \Rightarrow F \cup G \models X \to\to Y$;
$F \cup G \models X \to Y \Rightarrow F \cup G \models X \to\to Y$;
7) $F \cup G \models X \to\to Z \& F \cup G \models Y \to Z' \& Z' \subseteq Z \& Y \cap Z = \emptyset \Rightarrow F \cup G \models X \to Z'$.

FD or MVD φ is *syntactically* derived from the set of dependencies $F \cup G$ ($F \cup G \vdash_R \varphi$), if there is a finite sequence of FD or MVD $\varphi_1, \varphi_2, \ldots, \varphi_{m-1}, \varphi_m$, where $\varphi_m = \varphi$ and for all $\forall i = \overline{1, m-1}$ each φ_i is either the axiom of reflexivity (FD's or MVD's) or belongs to $F \cup G$ or is derived with some inference rule (complementation for MVD's, augmentation (for FD's or MVD's), transitivity (for FD's or MVD's), mixed inference rules for FD's and MVD's) from the previous in this sequence φ_j, φ_k, $j, k < i$.

As it has been stated above, let sequence $\varphi_1, \varphi_2, \ldots, \varphi_{m-1}, \varphi_m$ be called proof of φ from set of dependencies $F \cup G$.

Let there be given certain sets F and G of FD's and MVD's respectively.

Closure $[F \cup G]_R$ — is a set of all FD's and MVD's that are syntactically derived from $F \cup G$: $[F \cup G]_R \stackrel{def}{=} \{\varphi | F \cup G \vdash_R \varphi\}$.

Lemma 6. *The following properties are valid:*
1) $F \cup G \subseteq [F \cup G]$ *(increase);*
2) $[[F \cup G]] = [F \cup G]$ *(idempotency);*
3) $F' \cup G' \subseteq F \cup G \Rightarrow [F' \cup G'] \subseteq [F \cup G]$ *(monotonicity).*
4) $[F] \subseteq [F \cup G]$, $[G] \subseteq [F \cup G]$, $[F] \cup [G] \subseteq [F \cup G]$.

From the propositions 1-3 it follows that operator $F \cup G \mapsto [F \cup G]_R$ is the closure operator.

Closure $[X]_{F \cup G, R}$ of a set X (with respect to the set of dependencies $F \cup G$ and scheme R) is the family of all right parts of MVD's which are syntactically derived from the set $F \cup G$:

$$[X]_{F\cup G,R} \stackrel{def}{=} \{Y | X \to\to Y \in [F \cup G]_R\}.$$

Obviously, $[X]_{F\cup G,R} \neq \emptyset$ since, for example, $X \in [X]_{F\cup G,R}$, ($X \to\to X$, $X \to X$ are axioms of reflexivity); the latter statement can be strengthened: actually performed inclusion $2^X \subseteq [X]_{F\cup G,R}$, where 2^X — Boolean of a set X.

Let $[X]_F$ — closure of a set X with respect to the set of FD's F [8]. Note that by definition $[X]_F \subseteq R$.

Lemma 7. *The following properties are valid:*

1. $Y \subseteq [X]_F \Rightarrow Y \in [X]_{F\cup G,R}$;

2. $[X]_{F\cup G,R} = [[X]_F]_{F\cup G,R}$.

Proof. To prove proposition 1 we will construct a proof of MVD $X \to\to Y$ from set of dependences $F \cup G$. Really, we have:
1. Proof of FD $X \to [X]_F$ from F ([8], lemma 9);
2. $[X]_F \to Y$ (axiom of reflexivity for FD's; by assumption, $Y \subseteq [X]_F$);
3. $X \to Y$ (with 1 and 2 according to the rule of transitivity for FD);
4. $X \to\to Y$ (with 3 according to the rule of extension FD to MVD).

Thus, by definition of closure $[X]_{F\cup G,R}$ it follows $Y \in [X]_{F\cup G,R}$.

Let's prove proposition 2. Let $Y \in [X]_{F\cup G,R}$; let's show that $Y \in [[X]_F]_{F\cup G,R}$. By definition of closure $[X]_{F\cup G,R}$ there is proof of MVD $X \to\to Y$ from the set of dependencies $F \cup G$. Let's make a proof of MVD $[X]_F \to\to Y$ from $F \cup G$.
1. $[X]_F \to\to X$ (axiom of reflexivity for MVD's because $X \subseteq [X]_F$ according to [8, lemma 9]);
2. Proof of MVD $X \to\to Y$ from $F \cup G$ which exists by assumption;
3. $[X]_F \to\to Y \setminus X$ (with 1 and 2 according to the rule of transitivity for MVD's);
4. $[X]_F \to\to Y$ (with 3 according to the rule of augmentation for MVD's which can be obtained by simplification of the MVD $[X]_F \cup (X \cap Y) \to\to Y \setminus X \cup (X \cap Y)$; really, $Y \setminus X \cup (X \cap Y) = Y$; $[X]_F \cup (X \cap Y) = [X]_F$, because $X \cap Y \subseteq [X]_F$).

Thus, we have $Y \in [[X]_F]_{F\cup G,R}$.

Let now $Y \in [[X]_F]_{F\cup G,R}$; let's show that $Y \in [X]_{F\cup G,R}$. By definition of closure $[[X]_F]_{F\cup G,R}$ there is proof of MVD $[X]_F \to\to Y$

from the set of dependencies $F \cup G$. Let's make a proof of MVD $X \to\to Y$ from $F \cup G$.

1. Proof of FD $X \to [X]_F$ from F ([8], lemma 9);
2. $X \to\to [X]_F$ (with 1 according to the rule of extension FD to MVD);
3. Proof of MVD $[X]_F \to\to Y$ from set $F \cup G$ which exists by assumption;
4. $X \to\to Y \setminus [X]_F$ (from the latest MVD's in sequences of proof of items 2 and 3 according to the rule of transitivity for MVD's);
5. $[X]_F \to [X]_F \cap Y$ (axiom of reflexivity for FD's);
6. $X \to [X]_F \cap Y$ (with 1 and 5 according to the rule of transitivity for FD's);
7. $X \to\to [X]_F \cap Y$ (with 6 according to the rule of extension FD to MVD);
8. $X \to\to Y$ (with 4 and 7 according to the additional rule for MVD's we have MVD $X \to\to (Y \setminus [X]_F) \cup ([X]_F \cap Y)$; which has to be simplified).

Thus, $Y \in [X]_{F \cup G,R}$.$\square$

Observe that operator $X \mapsto [X]_{F \cup G,R}$ is not closure operator; it is based on the fact that this operator has no idempotency property (notion $[[X]_{F \cup G,R}]_{F \cup G,R}$ has no sense).

Basis $[X]^{bas}_{F \cup G,R}$ of a set X with respect to the set of dependencies $F \cup G$ and scheme R is subset of closure $[X]_{F \cup G,R}$, such that:

1. $\forall W (W \in [X]^{bas}_{F \cup G,R} \Rightarrow W \neq \emptyset$ (i.e., basis contains only nonempty sets of attributes);

2. $\forall W_i, W_j (W_i, W_j \in [X]^{bas}_{F \cup G,R}$ & $W_i \neq W_j \Rightarrow W_i \cap W_j = \emptyset)$ (i.e., sets of basis are pairwise disjoint);

3. $\forall Y (Y \in [X]_{F \cup G,R} \Rightarrow \exists \mathcal{T} (\mathcal{T} \subseteq [X]^{bas}_{F \cup G,R}$ & $\mathcal{T} - finite$ & $Y = \bigcup_{W \in \mathcal{T}} W)$ (i.e., each set of attributes from closure $[X]_{F \cup G,R}$ is equal to finite union of some sets from basis).

Lemma 8. *The following properties are valid:*

1. $\bigcup_{W \in [X]^{bas}_{F \cup G,R}} W = R$ *for* $X \subseteq R$ *(i.e. basic is partition of R);*

2. $A \in [X]_F \Rightarrow \{A\} \in [X]^{bas}_{F \cup G,R}$.

These lemmas are needed to establish the following main results.

4 Correctness and Completeness of Axiomatic for FD's and MVD's

Let φ — FD or MVD.

Statement 1 (Correctness of axiomatic for FD's and MVD's). *If dependency φ is syntactically derived from the set of dependencies $F \cup G$, then φ is derived semantically from $F \cup G$:*

$$F \cup G \vdash \varphi \Rightarrow F \cup G \models \varphi.$$

Proof. The proof is carried out by induction in the course of proving.

Basis. The length of proof is equal to 1. It means that dependency φ is either trivial (FD or MVD) or $\varphi \in F \cup G$. In all these cases (if φ is the trivial FD, then use corollary 1 from [8]; if φ is the trivial MVD, then use lemma 1, proposition 1) semantic succession $F \cup G \models \varphi$ takes place.

Inductive step. Let $\varphi_1, \varphi_2, \ldots, \varphi_{m-1}, \varphi_m$, $m \geq 2$ is a proof of dependency φ (FD or MVD) from set $F \cup G$. Let us consider all possible cases for last element of sequence φ_m, where φ_m — FD or MVD.

The case when φ_m is either trivial (FD or MVD) or $\varphi_m \in F \cup G$ we consider in a way analogous to that used in the basis of induction.

Let φ_m — FD which is deduced from certain FD φ_i, $i < m$ according to the rule of completion. It is obvious that $F \cup G \vdash \varphi_i$; by induction assumption we have $F \cup G \models \varphi_i$. It remains to use lemma 5, proposition 1.

Similar cases are considered, where:

– φ_m — FD that results from previous in this sequence of FD's according to the rule of transitivity (lemma 5, proposition 2 are used);

– φ_m — MVD that results from MVD φ_i, where $i < m$, according to the rules of complementation or augmentation (lemma 5, proposition 3 for the rule of complementation or lemma 5, proposition 4 for the rule of augmentation are used);

– φ_m — MVD that results from previous in this sequence of MVD's according to the rule of transitivity (lemma 5, proposition 5 are used);

– φ_m — MVD that results from FD φ_i, where $i < m$, according to the rule of extension of FD to MVD (lemma 5, proposition 6 are used);

– φ_m — FD that results from previous in this sequence of MVD and FD according to the mixed inference rule for FD's and MVD's (lemma 5, proposition 7 are used).□

Statement 2 (Completeness of axiomatic for FD's and MVD's). *If dependency φ is derived semantically from the set of dependencies $F \cup G$, then φ is syntactically derived from $F \cup G$ under the assumption $|R| \geq 2$ and $|D| \geq 2$:*

$$F \cup G \models \varphi \Rightarrow F \cup G \vdash \varphi.$$

Proof. We now turn to the idea of proof [4], which we reconstruct and complement. We will prove our statement by contradiction. Let the set $F \cup G$ and the dependency φ (FD or MVD) are such that $F \cup G \models \varphi$ is fulfilled, but $F \cup G \vdash \varphi$ is not valid, that is $\varphi \notin [F \cup G]$.

To show the contradiction with $F \cup G \models \varphi$, make such model of set $F \cup G$ that dependency φ is not valid.

Let us fix two distinct elements a and b in the universal domain. Let the set X is the left part of dependency φ (FD or MVD). Let the cover $[X]_F$ consists of attributes $A_1, A_2, \ldots, A_k$. According to property 2, lemma 8 for $i = \overline{1,k}$ we have $\{A_i\} \in [X]^{bas}_{F \cup G,R}$, that is basis $[X]^{bas}_{F \cup G,R}$ partitions the scheme R of cardinality n on the sets $\{A_1\} = W_1, \{A_2\} = W_2, \ldots, \{A_k\} = W_k, W_{k+1}, \ldots, W_m$, where $m \leq n$.

The table t is constructed as follows: the number of rows is 2^{m-k}; for all attributes $A \in [X]_F$ each row $s \in t$ takes values only from the set $\{a\}$, that is $s(A) = a$; at the sets W_i for $i = \overline{k+1,m}$, rows take values either only from the set $\{a\}$ or only from the set $\{b\}$ (see Table 1).

Table 1. Table t from the proof of Statement 2

$[X]_F$	W_{k+1}	W_{k+2}	...	W_{m-1}	W_m
a	a	a	...	a	a
a	a	a	...	a	b
...	...	...	...	...	...
a	b	b	...	b	b

Consider two possible cases for φ.

Let φ — FD of the form $X \to Y$ such that $X \to Y \notin [F \cup G]$. Let's show that FD $X \to Y$ is not valid on the table t. Since $X \to Y \notin [F \cup G]$, then from inclusion $[F] \subseteq [F \cup G]$ (lemma 6, property 4) it follows $X \to Y \notin [F]$. Hence we have $Y \not\subseteq [X]_F \subset R$ ([8], lemma 9, property 2), that is $Y \cap R \setminus [X]_F \neq \emptyset$. Therefore, for arbitrary attribute $A \in Y \cap R \setminus [X]_F$ there exist such rows s_1 and s_2, that $s_1(A) = a$ and $s_2(A) = b$ (by construction of table t), hence, $s_1|Y \neq s_2|Y$. On account of the equality $s_1|X = s_2|X$ (by construction of table t) we have $(X \to Y)(t) = false$.

Let φ — MVD of the form $X \to\to Y$ such that $X \to\to Y \notin [F \cup G]$. Let's show that MVD $X \to\to Y$ is not valid on the table t. By assumption $X \to\to Y \notin [F \cup G]$, it follows:

1. $Y \not\subseteq [X]_F$, because then the MVD $X \to\to Y$ will have a proof (and hence will belong to the set $[F \cup G]$):

a. Proof of FD $X \to [X]_F$ from the set F, and therefore from the set $F \cup G$ ([8], lemma 9);

b. $[X]_F \to Y$ (axiom of reflexivity for FD's; recall that by assumption $Y \subseteq [X]_F$);

c. $X \to Y$ (with a and b according to the rule of transitivity for FD's);

d. $X \to\to Y$ (with c according to the rule of extension FD to MVD);

2. $Y \neq \bigcup W_i$ for some $1 \leq i \leq m$ because $W_i \in [X]^{bas}_{F \cup G, R}$, that is MVD $X \to\to \bigcup W_i \in [F \cup G]$;

3. Note also that $Y \neq \emptyset$ because MVD $X \to\to \emptyset$ is the axiom of reflexivity and it belongs to $[F \cup G]$;

4. It still remains to consider the case where $Y \cap W_i \subset W_i$, $Y \cap W_i \neq \emptyset$ for some $k + 1 \leq i \leq m$. Suppose that MVD $X \to\to Y$ holds at the table t. Since for arbitrary rows s_1 and s_2 the equality $s_1|X = s_2|X$ is fulfilled (by construction of table t), then for fixed i we choose s_1 and s_2 as follows: $range(s_1|W_i) = \{a\}$ and $range(s_2|W_i) = \{b\}$. According to the rules of decomposition (item 2) we have $\{X \to\to W_i, X \to\to Y\} \vdash X \to\to W_i \setminus Y$; it follows that there exists row s_3, that $s_3|W_i = s_1|(W_i \setminus Y) \cup s_2|(W_i \cap Y)$, that is $s_3|W_i$ takes values from both sets $\{a\}$ and $\{b\}$; this contradicts the construction of the table t.

Thus, MVD $X \to\to Y$ which does not belong to the set $[F \cup G]$, is not valid on the table t.

Let's now show that table t is the model of set $F \cup G$. Consider two possible cases.

I. Given FD $U \to Z \in F \subseteq F \cup G$. We will show that $(U \to Z)(t) = true$, that is for arbitrary rows s_1 and s_2 the implication $s_1|U = s_2|U \Rightarrow s_1|Z = s_2|Z$ is valid.

There are two possible cases for the set of attributes U.

Case 1: $U \cap R \setminus [X]_F = \emptyset$ that is $U \subseteq [X]_F$. Then for arbitrary rows s_1 and s_2 by construction of the table t we have $s_1|U = s_2|U$; so we need to show the equality $s_1|Z = s_2|Z$. To prove this, it is sufficient to make sure that $Z \subseteq [X]_F$. For this purpose we consider the following proof of FD $X \to Z$ from the set F:

1. Proof of FD $X \to [X]_F$ from the set F ([8], lemma 9);
2. $[X]_F \to U$ (axiom of reflexivity for FD's; recall that by assumption $U \subseteq [X]_F$);
3. $X \to U$ (the rule of transitivity is applied to the FD $X \to [X]_F$, which is the last element of proof 1 and FD $[X]_F \to U$ 2);
4. $U \to Z$ (element of set F);
5. $X \to Z$ (with 3 and 4 according to the rule of transitivity for FD's).

Hence we have $F \vdash X \to Z$, that is $Z \subseteq [X]_F$; it follows that $s_1|Z = s_2|Z$.

Case 2: $U \cap R \setminus [X]_F \neq \emptyset$. In the case when $Z \subseteq [X]_F$, FD $U \to Z$ is valid trivially on account of the construction of the table t.

Let $Z \nsubseteq [X]_F$. We first show that FD $U \to Z$, where $Z \cap W_i \neq \emptyset$ and at that $U \cap W_i = \emptyset$, for $k+1 \leq i \leq m$, does not belong to the set F. Assume the contrary. Then there exists proof for FD, which is not valid on the table t:

1. $U \to Z$ (element of set F);
2. $Z \to Z \cap W_i$ (axiom of reflexivity for FD's);
3. $U \to Z \cap W_i$ (with 1 and 2 according to the rule of transitivity for FD's);
4. $Z \cap W_i \to W_i$ (by construction of the table t; recall that $\forall A', A'' \in W_i(s(A') = s(A''))$);
5. $U \to W_i$ (with 3 and 4 according to the rule of transitivity for FD's);

6. $X \to\to W_i$ (by construction of the table t; recall that $W_i \in [X]^{bas}_{F\cup G,R}$);
7. $X \to W_i$ (with 6 and 5 according to the mixed inference rule for FD's and MVD's; recall that by assumption $U \cap W_i = \emptyset$).

Thus, for some i, $k+1 \le i \le m$ we have the proof of FD $X \to W_i$, which is not valid on the table t (by construction of the table t). Therefore, FD $U \to Z$ where $Z \cap W_i \ne \emptyset$ and at that $U \cap W_i = \emptyset$ does not belong to the set F.

Considering that $\bigcup_{W_i \in [X]^{bas}_{F\cup G,R}} W_i = R$ (property 1, lemma 8) we write the set Z in the form $Z = \bigcup_{i=1}^{m}(Z \cap W_i)$. Fix i and show that FD $U \to Z \cap W_i$ is valid on the table t. Let's consider all possible cases:
1. if $Z \cap W_i \subseteq [X]_F$, then FD $U \to Z \cap W_i$ is valid on the table t (by construction);
2. if $Z \cap W_i \subseteq W_i$ for $k+1 \le i \le m$ then $U \cap W_i \ne \emptyset$ as it has been showed. Let's make the proof of FD $U \to Z \cap W_i$:

a. $U \to U \cap W_i$ (axiom of reflexivity for FD's);

b. $U \cap W_i \to Z \cap W_i$ (by construction of the table t; recall that $\forall A', A'' \in W_i (s(A') = s(A'')$ and on account of the inclusions $U \cap W_i \subseteq W_i$, $Z \cap W_i \subseteq W_i$);

c. $U \to Z \cap W_i$ (with a and b according to the rule of transitivity for FD's).

Thus, FD $U \to Z \cap W_i$, $1 \le i \le m$, is valid on the table t; hence FD $U \to \bigcup_{i=1}^{m}(Z \cap W_i)$ is valid [8, lemma 7, conclusion 6], therefore FD $U \to Z$ is valid.

II. Let's consider MVD $U \to\to Z \in G \subseteq F \cup G$ and show that $(U \to\to Z)(t) = true$.

We first show that MVD $U \to\to Z$, where $Z \cap W_i \subset W_i$ ($Z \cap W_i \ne \emptyset$) and at that $U \cap W_i = \emptyset$ for $k+1 \le i \le m$, does not belong to the set G. Assume the contrary. Then there exists proof for MVD, which is not valid on the table t:
1. $U \to\to Z$ (element of set G);
2. $R \backslash W_i \to\to Z$ (from inclusion $R \backslash W_i \supseteq Z \backslash W_i$ and with 1 according to the rule of augmentation for MVD's we have $U \cup R \backslash W_i \to\to Z \cup Z \backslash W_i$; it remains to consider that $U \cap W_i = \emptyset$);

3. $X \to\to W_i$ (by construction of the table t; recall that $W_i \in [X]^{bas}_{F\cup G,R}$);
4. $X \to\to R \setminus W_i$ (with 3 according to the rule of difference (item 3));
5. $X \to\to Z \cap W_i$ (with 4 and 2 according to the rule of transitivity for MVD's we have $X \to\to Z \setminus (R \setminus W_i)$, which should be simplified).

Considering that $Z \cap W_i \subset W_i$ $(Z \cap W_i \neq \emptyset)$ we have contradiction with assumption that W_i belongs to basis $[X]^{bas}_{F\cup G,R}$. Thus, MVD $U \to\to Z$, where $Z \cap W_i \subset W_i$ $(Z \cap W_i \neq \emptyset)$ and at that $U \cap W_i = \emptyset$ for $k+1 \leq i \leq m$, does not belong to the set G.

On account of the property $\bigcup_{W_i \in [X]^{bas}_{F\cup G,R}} W_i = R$ (property 1, lemma 8) write the set Z in the form $Z = \bigcup_{i=1}^{m}(Z \cap W_i)$. Fix i and show that MVD $U \to\to Z \cap W_i$ is valid on the table t. Let's consider all possible cases for $Z \cap W_i$:
1. $Z \cap W_i = \emptyset$; then $U \to\to \emptyset$ is an axiom of reflexivity and is valid trivially;
2. $Z \cap W_i = W_i$; then $U \to\to W_i$ is valid by construction of the table t (recall that table t consists of combinations of all possible values on the sets of attributes W_i and $R \setminus W_i$, $k+1 \leq i \leq m$);
3. $Z \cap W_i \subseteq [X]_F$; then $U \to\to Z \cap W_i$ holds true by construction of table t;
4. $Z \cap W_i \subset W_i$ for $k+1 \leq i \leq m$ and $Z \cap W_i \neq \emptyset$, then $U \cap W_i \neq \emptyset$ as it has been showed.

Let's show that $U \to\to Z \cap W_i$ is valid for this case; that is for arbitrary rows s_1 and s_2 such that $s_1|U = s_2|U$, there exists row s_3 that $s_3 = s_1|U \cup s_1|(Z \cap W_i) \cup s_2|R \setminus (U \cup (Z \cap W_i))$. By conditions $s_1|U = s_2|U$ and $U \cap W_i \neq \emptyset$, it follows equality $s_1|W_i = s_2|W_i$ (by construction of the table t). Restrict both parts of this equality to the set Z: $(s_1|W_i)|Z = (s_2|W_i)|Z$. According to the property of restriction operator $((U|Y)|Z = (U|(Y \cap Z)$ [3, p. 24]) it follows $(s_1|(Z \cap W_i) = s_2|(Z \cap W_i)$. Thus, $s_3 = s_2|U \cup s_2|(Z \cap W_i) \cup s_2|R \setminus (U \cup (Z \cap W_i)) = s_2 \in t$. Consequently, MVD $U \to\to Z \cap W_i$ is valid on table t.

From the above and by lemma 3, item 1 it follows $\{U \to\to Z \cap W_1, \ldots, U \to\to Z \cap W_k\} \vdash U \to\to Z$. Therefore, MVD $U \to\to Z$ is valid on table t.□

Conditions $|R| \geq 2$ and $|D| \geq 2$ are obtained through a detailed

analysis of the proofs.

Theorem 1. *The relations of semantic and syntactic succession coincide for axiomatic of FD's and MVD's under the assumption* $|R| \geq 2$ *and* $|D| \geq 2$*:*

$$F \cup G \models \varphi \Leftrightarrow F \cup G \vdash \varphi.$$

The proof follows directly from Statements 1 and 2 (Section 4).

Analogous theorem holds for axiomatic of MVD's (for axiomatic of FD's see [8]).

5 Conclusion

In this paper we have constructed a fragment of the mathematical theory of normalization in table databases — axiomatics for multivalued dependencies and axiomatics for functional and multivalued dependencies are considered. In particular, it was produced the proof of correctness of these axiomatics and completely reconstructed the known in the literature proof of the completeness.

References

[1] H. Darwen, C. Date, R. Fagin. *A Normal Form for Preventing Redundant Tuples in Relational Databases.* 15th International Conference on Database Theory ICDT2012 Proc. Berlin, Germany, March 26–30, 2012, pp. 114–126.

[2] A. Bahmani, M. Naghibzadeh, B. Bahmani. *Automatic database normalization and primary key generation.* CCECE/CCGEI Proc. Niagara Falls, Canada, May 5-7, 2008, pp. 11–16.

[3] V. N. Redko, Yu.Y. Brona, D.B. Bui, S.A. Polyakov. *Relational Data Bases: Table Algebras and SQL-like Languages.* Akademperiodyka, Kyiv, 2001 (in Ukrainian).

[4] C. Beeri, R. Fagin, J. Howard. *A complete axiomatization for functional and multivalued dependencies.* ACM-SIGMOD Conference,Toronto, Canada, Aug. 3–5, 1977, pp. 47–61.

[5] D. Bui, A. Puzikova. *Axiomatics for multivalued dependencies in table databases: correctness, completeness, completeness criteria.* 5th CrISS-DESSERT Workshop,Brunow, Poland, June 29 - July 3, 2015.

[6] R. Lyndon. *Notes on Logic.* Van Nostrand Company, Inc., Princeton, 1966.

[7] L. A. Skornyakov. *Elements of the theory of structures.* Nauka, Moscow, 1982 (in Russian).

[8] D. B. Bui, A. V. Puzikova. *Completeness of Armstrongs axiomatic.* Bulletin of Taras Shevchenko National University of Kyiv. Series Physics and Mathematics, no. 3 (2011), pp. 103–108.

Dmitriy Bui, Anna Puzikova, Received June 30, 2015

Dmitriy Bui
Institution: Taras Shevchenko National University of Kyiv
Address: 64/13, Volodymyrska Street, City of Kyiv, Ukraine
Phone: +380505548802
E–mail: buy@unicyb.kiev.ua

Anna Puzikova
Institution: Taras Shevchenko National University of Kyiv
Address: 64/13, Volodymyrska Street, City of Kyiv, Ukraine
Phone:+380662748725
E–mail: anna_inf@mail.ru

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

A Prediction System Based on Fuzzy Logic

Sergiu Chilat

Abstract

This paper presents a new way of implementing the forecasting process using a neural network and a fuzzy controller. At the first stage of the process, the neural network performs the forecasting based on internal parameters, and then the fuzzy controller adapts the results to the changes of the external parameters. The main problem of neural networks in forecasting is the impossibility to foresee the changes in the external parameters. This problem is solved by the fuzzy controller, which reacts to any change in the external parameters and adjusts the forecasted data provided the neural network.

Keywords: neural network, fuzzy logic, fuzzy controller, rule base, time series, diagnostication.

1 Introduction

Forecasting is the process of making statements about the future outcomes of events, processes etc., based on the analysis of the circumstances that determine their occurrence and evolving. The anticipation of future values of certain dimensions is a widely popular challenge due to its importance in multiple fields like medical science, tech, economy etc.

The complexity of the process of discrete sequences forecasting is high due to the fact that, unlike the algorithmically well-organized interpolation procedures, the forecasting needs extrapolation of data from the past towards the future. Moreover, many unfamiliar factors that influence the discrete sequences must be considered. There are many studies dedicated to the elaboration of the mathematical model

of forecasting. Most of them are based on probabilistic and statistical tools, and their use requires a considerable amount of experimental data, which is not always available or even possible to accumulate.

In the last years, a growing interest in neural networks in addressing forecasting problems can be observed. These are considered a most close model to reproduce human brain activity, and can be taught to identify regularities previously unobserved.

In the fuzzy specialty literature, the forecasting methods are divided in 4 groups:

- based on actual data, presented as temporary strings;
- based on heuristic information, obtained from highly-qualified specialists in the field;
- based on mathematical, biological or historical analogies;
- complex methods – combine multiple approaches [1].

In this paper a model for sales forecasting will be presented, being created based on a neural network and a fuzzy controller connected serially. At the first stage, the neural network will generate the forecasting data based on the data from previous sales (internal parameters). At the second stage, the data generated by the neural network will be automatically adapted to the influence of external factors (parameters).

Thus, forecasting is a continuous adaptive process, which needs its results to be modified in order to have optimal overall results and taking into consideration every change in the external factors.

2 Forecasting method

The determination of objectives is one of the main purposes of forecasting. The objectives selection is performed as a result of the problem analysis. After the selection a tree of objectives classified by index of priority is created (Fig. 1).

Another object of the study is the method of creation of forecasting systems which are based on up-to-date intellectual technologies:

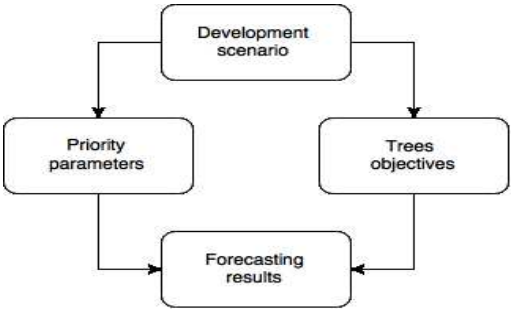


Figure 1. Forecasting process

fuzzy set theory, neural networks, fuzzy logic methods and genetic algorithms.

The most important issues in the forecasting models creations are the following:

- there are some unclear links between the internal and external parameters (fuzzy);
- there is no statistical stability in neither the input nor the output parameters [2, 3].

Thus, the use of classical methods like extrapolation and regressive analysis are not useful, because their main disadvantage is the need of a big amount of statistical data for many parameters and for different time spans.

Fuzzy sets offer the possibility to formalize values, to determine cause-and-effect relationships between parameters and the factors that may influence them, and to formulate a prognosis in a state of uncertainty.

In Fig.2 there is the diagram of the forecasting method which is based on fuzzy logic methods.

The algorithm to create the forecasting mathematical model is:

- establish input and output parameters;

- based on an expert system, form fuzzy sets and select the unit of measurement;
- define the set membership functions;
- form the rules base;
- establish the decision making algorithm [4].

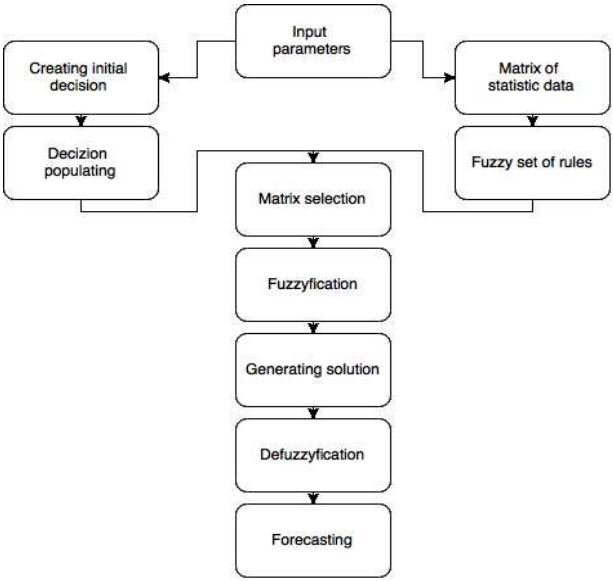


Figure 2. Forecasting method diagram

While building the mathematical algorithm, the object is investigated as the function (1):

$$P = FuzzyPrognosis(t_1, t_2, t_n) \tag{1}$$

where n – the number of terms (input parameters), P – the final solution.

The working algorithm of the *FuzzyPrognosis* function is:

1. collecting the input parameters set that can influence the final solution;
2. generate linguistic variables for each of the input parameters and of the respective correspondences;
3. create a graph that will reflect the parameter classification;
4. collect statistical data;
5. fuzzify data (transform the precise values of input parameters into linguistic fuzzy values);
6. the membership functions of the linguistic variables are given in the following form (2):

$$fA(x) = \frac{1}{1 + \sum_{i=1}^{n} \left[\frac{x - P_i}{P_i}\right]^2} \tag{2}$$

   where P_i are the configuration parameters;
7. get the modelling results; it is done by transforming the fuzzy sets back into exact values (defuzzification).

The configuration of the fuzzy model is made by the completion of the rules base and composing the membership function (2) and is represented by the form in Fig. 3.

The rule base consists of data received from experts, in a cause-effect format(3).

$$S : \begin{cases} S_1 & \text{if} \quad A_1 \quad \text{then} \quad B_1 \\ S_2 & \text{if} \quad A_2 \quad \text{then} \quad B_2 \\ & \quad \ldots \\ S_n & \text{if} \quad A_n \quad \text{then} \quad B_n \end{cases} \tag{3}$$

The matrix generation (3) takes place using genetic algorithms approaches: as a result of a finite number of iterations of the genetic

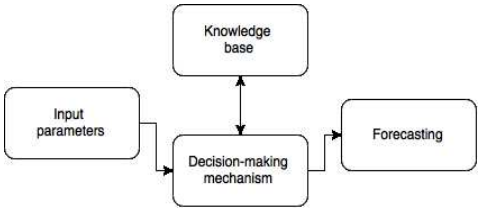


Figure 3. The fuzzy decision making model

algorithm a solution is chosen (chromosome) having the maximal quality index, based on the initially announced criteria.

Every iteration of the genetic algorithm consists of several consecutive operations:

- initializing the set of chromosomes;
- creating the chromosome descendants as a result of combining base chromosomes with certain mutations;
- calculating the quality level for the newly created chromosome.

In order to determine the subject of the prognosis, the analysed and forecasted variables must be specified. It is very important to know the level of particularization, which is influenced by many factors: availability and quality of data, the costs of the analysis of user preferences in the system.

In cases where the best variable combination isn't clear, different alternatives could be tried, selecting the one with the best results.

Another important step in building a neural networks based forecasting system is the definition of the following three parameters: the period, the interval and the horizon of the forecasting.

The *forecasting period* is the essential time unit for which the prognosis is made.

The *forecasting horizon* represents the number of periods in future for which the prognosis is made. For example, a forecasting for a period

of one week could be necessary, presenting separate data for each day. In this case, the period is one day, while the horizon – one week.

The *forecasting interval* – the frequency of new prognosis.

The coherent selection during a particular period or horizon is the most difficult part of neural network based forecasting. The accuracy of a prognosis on a particular problem, has a solid impact on the whole forecasting system. Also, of a big influence on the prognosis accuracy is the data set chosen for the network to learn [5].

The number of output nodes is relatively easy to specify as it is directly related to the problem under study. For a time series forecasting problem, the number of output nodes often corresponds to the forecasting horizon. There are two types of forecasting: one-step-ahead (which uses one output node) and multi-step-ahead forecasting. Two ways of making multi-step forecasts are reported in the literature. The first is called the iterative forecasting as used in the Box-Jenkins model in which the forecast values are iteratively used as inputs for the next forecasts. In this case, only one output node is necessary. The second called the direct method is to let the neural network have several output nodes to directly forecast each step into the future.

3 The issue

A specific example must be investigated: forecasting the amount of sales of an enterprise which sells food. The medium is nondeterministic, because the conventional models do not allow determining with certitude what will happen in the next moment in time and because all factors that could possibly influence the moment are not known. There are several financial signs [6]:

Enterprise activity

- sales history (amount of products, amount of money);
- storage operations history;
- advertising activity indicators.

External factors

- prices for competitors services;
- market state;
- inflation level;
- currency exchange state;
- stock market index.

There are also secondary factors (Table 1), which can also influence sales. These parameters have different significance, different values and origins.

Table 1. Secondary factors

Parameter	**Importance**
P1 = Products is store	I(P1) = 30
P2 = Advertising intensity	I(P2) = 20
P3 = Raw material price	I(P3) = 60
P4 = Geographical position	I(P4) = 10
P5 = Inflation	I(P5) = 40
P6 = Competitors prices	I(P6) = 30
P7 = Import	I(P7) = 10
P8 = Market share	I(P8) = 30
P9 = Customers concentration	I(P9) = 10
P10 = Monetary exchange	I(P10) = 20

As a consequence of the analysis made, it can be easily observed that some parameters cannot be included in the model because of the impossibility to obtain truthful data for them, while other ones do not have an influence on the model dynamics big enough to be considered and can be excluded from the model without significant loss in precision. It should be mentioned that the sales history can provide about 50-60% of the total amount of information necessary for the neural network.

The issue with the prognosis of sales for a company has some characteristics that make the neural networks an efficient solution to pick:

- the amount of data can be relatively small (data regarding sales of new products);
- there can be blank data in the database;
- data can be distorted;
- an adaptation mechanism of the model to newly incoming data is necessary;
- it is quite difficult to create a linear algebra model for this case;
- the number of products can be big [7, 8].

4 Forecasting algorithm

Table 2 contains data regarding sales history (in thousands of units) during the time between 01.01.2013 and 31.12.2014, grouped by months. The data should be used as input for the neural network.

Next step is training. The most popularly used training method is the backpropagation algorithm which is essentially a gradient steepest descent method. For the gradient descent algorithm, a step size, which is called the learning rate in ANNs literature, must be specified. The learning rate is crucial for backpropagation learning algorithm since it determines the magnitude of weight changes. It is well known that the steepest descent suffers the problems of slow convergence, inefficiency, and lack of robustness. After obtaining results from neural network and comparing with statistical data, the following data is obtained (Fig. 4 time interval 01.2014 12.2014).

The maximum fault value of the network is of 0,3 units (6%), while the medium fault is of 0,04 units (0,8%), which means it can be used in forecasting.

The next step is the generation of the forecasting data for the first 6 months of the year 2015. Using the neural network, the following results are obtained (Table 3).

Table 2. Secondary factors

Month	**Sales**	**Month**	**Sales**
01.2013	2.0	01.2014	2.3
02.2013	2.5	02.2014	3.1
03.2013	2.5	03.2014	3.2
04.2013	2.7	04.2014	3.5
05.2013	3.2	05.2014	4.0
06.2013	3.7	06.2014	4.3
07.2013	3.8	07.2014	3.5
08.2013	3.9	08.2014	4.3
09.2013	2.8	09.2014	3.3
10.2013	2.4	10.2014	2.5
11.2013	3.8	11.2014	4.3
12.2013	4.5	12.2014	4.8

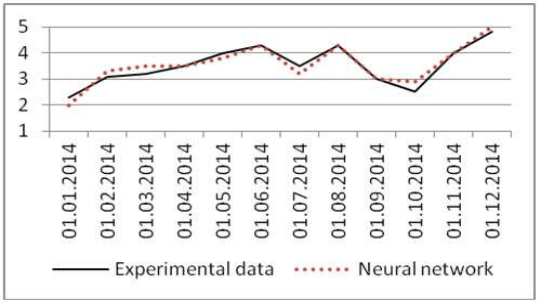


Figure 4. Verification sample

Table 3. Forecasting results

Month	Sales
01.2015	2.35
02.2015	3.0
03.2015	3.1
04.2015	3.7
05.2015	4.2
06.2015	4.7

5 Adapting results to external factors

The data obtained is generated solely based on internal parameters. But in real world, the examined results can be influenced by external factors, too (Table 1), which are dynamic and mostly unpredictable.

Thus, there appears the problem of accuracy (the data obtained previously can only be applied to cases that exist in isolation from external factors – exterior has a null impact or should be ignored for other reasons). But, in reality, these factors cannot be ignored, so, the forecasting results (Table 3) will be inapplicable.

In order to solve this problem, the present article suggests using a fuzzy logic controller that receives data given by the neural network (Table 3) and values of external parameters (Table 1) and adapts data for daily use.

At the first stage, the rule base for the controller will be composed (using Table 1).

For parameter P1 – Products in store:

- IF *products in store = few* THEN *sales decrease by I(P1) percent*
- IF *products in store = sufficient* OR *products in store = many* THEN *sales remain at the prognosed level*

For parameter P2 – Advertisement intensity:

- IF *advertisement = none* THEN *sales decrease by I(P2) percent*

- IF *advertisement = few* THEN *sales decrease by I(P2)/2 percent*
- IF *advertisement = much* THEN *sales increase by I(P2)/2 percent*

For parameter P3 – Raw material price:

- IF *raw material price = high* THEN *sales decrease by I(P3) percent*
- IF *raw material price = low* THEN *sales increase by I(P3)/2 percent*

For parameter P5 – Inflation:

- IF *inflation = high* THEN *sales decrease by I(P5) percent*
- IF *inflation = low* THEN *sales increase by I(P5)/2 percent*

For parameter P6 – Competitors prices:

- IF *competitors prices = high* THEN *sales increase by I(P6)/3 percent*
- IF *competitors prices = low* THEN *sales decrease by I(P3) percent*

For parameter P8 – Market share:

- IF *market share = high* THEN *sales increase by I(P8)/4 percent*
- IF *market share = low* THEN *sales decrease by I(P8) percent*

One can modify the number of the parameters or the rules base in order to increase the controller efficiency.

Due to the fact that the adaptation process is continuous, the application of the fuzzy controller will increase the forecasting precision, because both the predictable and random factors will be taken into account.

6 Conclusions

Using an intelligent system based on neural networks, a sales-related forecasting can be generated, based on statistical data describing sales in the past. One of the most important advantages of the method is the lack of the need to have a strict specification for the mathematical model.

When forecasting for a period of several months, it is quite difficult (even when using neural networks) to predict external factors of influence.

In order to facilitate this, an innovative method has been suggested creating intelligent systems that incorporate neural networks together with a fuzzy controller (connected serially), which implies dividing the process into 2 stages:

1. Forecasting – the neural network generates temporary strings based on internal parameters (sales history for the past 2 years);

2. Adaptation to external factors – the fuzzy controller adapts the data produced by the neural network to certain external factors.

As a consequence, the main problem of the neural networks in forecasting is the impossibility to foresee the changes in external parameters, which is resolved by the fuzzy controller, which reacts to every change in the external parameters and adjusts the forecasting data accordingly.

References

[1] D. Srinivasan, S.S. Tan, C.S. Chang, E.K. Chan. *Practical implementation of a hybrid fuzzy neural network for one-day-ahead load forecasting.* IEE Proc. Gener. Transm. Distrib. 1998. Vol. 145. 6.

[2] G.E.P. Box, G.M. Jenkins. *Time series analysis: Forecasting and control*, San Francisco: Holden-Day, 1970.

[3] M.H. Hassoun. *Fundamentals of Artificial Neural Networks*, A Bradford book, The MIT Press Cambridge Massachusetts, 1995.

[4] N. Morariu, E. Iancu, S. Vlad. *A neural network model for time series forecasting.* Romanian Journal of Economic Forecasting. 2009, No. 4. pp. 213 - 223.

[5] S. Mahfoud, G. Mani. *Financial Forecasting Using Genetic Algorithms.* Applied Artificial Intelligence. 1996, Vol. 10, No.6. pp. 543 – 560.

[6] Basaran Filik U., Kurban M. *A New Approach for the Short-Term Load Forecasting with Autoregressive and Artificial Neural Network Models.* International Journal of Computational Intelligence Research. 2007, No.3. pp. 66 – 71.

[7] W. Charytoniuk, M.S. Chen. *Short-term Forecasting in Power Systems Using a General Regression Neural Network.* IEEE Trans. on Power Systems. 1995. Vol. 7. 1.

[8] A. Garch. *Forecasting Model to Predict Day-Ahead Electricity Prices R.C. Garcia [at al.].* IEEE Transactions on Power Systems. 2005, Vol. 20, No. 2. pp. 867 – 870.

Sergiu Chilat

Received July 12, 2015

Technical University of Moldova
Bd. Ştefan cel Mare, 168, MD-2004, Moldova, Chişinău
Phone: +37360333284
E–mail: chilatsergiu@gmail.com

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

On preservation of keys in table algebra

Aleksey Senchenko

Abstract

The problem of invariance of keys, including candidate keys, with respect to operations of table algebras (a modern analog of classical relational Codd's algebra) is investigated. It is shown that keys are invariant with respect to operations of intersection, difference, selection, join and division, but for candidate keys invariancy isn't carried out. It is shown that keys, including candidate keys, are invariant with respect to operation of renaming. The necessary and sufficient conditions under which the keys, including candidate keys, are invariant with respect to operations of projection and the active complement are found. The results of work represent theoretical and practical interest and can be used for a choice of optimum keys at design of relational databases.

Keywords: database, table algebra, key, candidate key.

1 Introduction

Currently, databases are widely used in almost all areas of human activity. For all the variety of different types of databases the most common are relational (table) databases, mathematical model of which was first proposed by E. Codd [1, 2]. From mathematical point of view, a relational database is a finite set of finite relations between different predefined sets of basic data. Table algebra introduced by V.N. Red'ko and D.B. Buy [3], is based on Codd's relational algebra and significantly develop them. They formed the theoretical foundation of modern query language databases. Elements of the carrier of table algebra specify relational data structures, and signature operations are based on the basic table manipulation in relational algebra and SQL-like languages.

In relational databases the keys of the table are very important – one or more of their attributes, values of which uniquely identify the table entry. The keys (primary and foreign) establish binary relation such as "one-to-many". These links are used to maintain the integrity of the database. Typically, the keys are defined in such a way that they are invariant to any changes in database records. In this paper we consider the question of invariance of the keys with respect to signature operations of table algebra. The results are of interest to select the optimal key in the design of databases [4, 5].

2 Basic definitions

We fix a non-empty set of attributes $\mathfrak{R} = \{A_1, \ldots, A_n\}$. An arbitrary finite subset of $\mathfrak{R}$ is called schema, wherein the schema can be an empty set. A string (tuple) s of schema R is the set of pairs $s = \{(A'_1, d_1), \ldots, (A'_k, d_k)\}$, whose projection on the first component is equal to R, and attributes $A'_1, \ldots, A'_k$ are distinct, i.e., string is a functional binary relation. A table of schema R is a finite set of strings of schema R, the number of strings in the table T is denoted by $|T|$. Let T_1 and T_2 are tables of schema R. On the set of tables of schema R we introduce some operations as follows:

1) the union $\bigcup\limits_R$ of two tables of schema R – is a table consisting of those and only those strings, that belong to at least one of the source tables;

2) the intersection $\bigcap\limits_R$ of two tables of schema R – is a table consisting of those and only those strings, that belong to both of the source tables;

3) the difference $T_1 \underset{R}{-} T_2$ between the two tables of schema R – is a table consisting of those and only those strings, that belong to the table T_1 and do not belong to the table T_2.

The introduction of the saturation operation requires one auxiliary concept. Active domain of attribute A with respect to the table T is a set $D_{A,T} = \{d | \exists s \in T \wedge (A, d) \in s\}$, consisting, saying meaningfully, all possible values of attribute A in the strings of the table T (if the

attribute is not included in the table schema, it's active domain is empty). Attributes of the table, the power of active domains of which is greater than one, are said to be multi-valued, otherwise the attributes are called one-valued. The saturation $C(T)$ is a table $\prod\limits_{A\in R} D_{A,T}$, where R – is the schema of the table T, and $\prod$ – is an operator of direct (Cartesian) product corresponding to indexing $A \mapsto D_{A,T}$, $A \in R$. The active complement of table T is a table $\tilde{T} = C(T) \underset{R}{-} T$.

We introduce the definition of the projection operation. The projection with respect to the set of attributes $X \subseteq R$ is called unary parametric operation π_X, the value of which is a table consisting of restrictions on X of all the strings of the original table: $\pi_X(T) = \{s \mid x \mid s \in T\}$. The definition of the restriction is standard: $s \mid x = s \bigcap X \times pr_2 s$, where $pr_2 s$ – projection of string s by the second component. Any restriction s' of string s we call the substring, and it is obvious that $s' \subseteq s$.

We introduce the definition of the selection operation. Unary parametric operation σ_P, that compares the table to its subtable containing the strings for which the predicate P has the true value, is called selection by predicate $P : S \to \{true, false\}$, where S – the set of all strings.

The introduction of the join operation requires one auxiliary concept. Binary relations ρ and τ are called consistent (denoted $\rho \approx \tau$), if $\rho \mid X = \tau \mid X$, where $X = pr_1\rho \bigcap pr_1\tau$ [6]. The join is called a binary operation $\otimes$, value of which is a table consisting of all unions of consistent strings of source tables, i.e., $T_1 \otimes T_2 = \{s_1 \bigcup s_2 \mid s_1 \in T_1 \wedge s_2 \in T_2 \wedge s_1 \approx s_2\}$. For a table T with the set $O = \{O_1, \ldots, O_z\}$ of all of the one-valued attributes and the values of their active domains $D_{O_1,T} = \{o_1\}, \ldots, D_{O_z,T} = \{o_z\}$ it is obvious that $T = \pi_Y(T) \otimes T'$, where $Y = R - O$ and $T' = \begin{matrix} O_1 & \cdots & O_z \\ o_1 & \ldots & o_z \end{matrix}$, i.e., $T' = \{\{(O_i, o_i)\} \quad i = \overline{1, z}\}$.

We introduce the definition of the division operation of two tables. Let T_1 – the table of schema R_1, T_2 – the table of schema R_2 and $R_2 \subseteq R_1$. The division of the table T_1 for the table T_2 is such table of

schema $R_1 - R_2$: $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2 = \{s \in \pi_{R_1-R_2}(T_1) \mid \{s\} \otimes T_2 \subseteq T_1\}$. If such strings s in the table $\pi_{R_1-R_2}(T_1)$ do not exist, by definition in this case $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2 = T_\emptyset = \emptyset$.

We introduce the definition of operation renaming the attributes. The renaming is called unary, generally partial operation $R\,T_\xi$, where ξ – is injective function on the set of attributes. This operation is carried out only by renaming the attributes of tables in accordance with the mapping parameter ξ. Informally speaking, renaming of table is reduced to renaming of the first component of couples – elements of strings.

Table algebra is called a partial algebra with the carrier - the set of all tables of an arbitrary schema - and signature, which contains above nine operations (saturation is seen as supporting the operation). The table algebra identifies two special tables: the table $T_\varepsilon = \{\varepsilon\}$, where ε – empty string, and the schema of the table T_ε is the empty set, and the table $T_\emptyset = \emptyset$ – empty set of arbitrary strings (including a non-empty schema). Table T, which is not a special one, is called unsaturated if the inequality $\tilde{T} \neq T_\emptyset$ (obviously, in this case $T \neq C(T)$) holds.

The attribute set $K \subseteq R$ is called the key of table T, if for any strings $s_1, s_2 \in T$ the implication $s_1 \mid K = s_2 \mid K \to s_1 = s_2$ is performed; in other words, restrictions on key attributes of all the strings of the table T are distinct. The key K is called a candidate key of the table T, if all its proper subsets are not the keys of T. It is easy to see that the schema of a table will be it's key, so the most interesting are the so-called non-trivial keys that are proper subset of the table schema. From the definition of the key it implies that the key of table $T_\emptyset$ is any subset of the attributes of its schemes, and a candidate key – the empty set. For a table of T_ε key (including candidate) is empty set. In practice, in the actual database, the special tables are used extremely rare (especially table T_ε), so for the sake of convenience the results only in non-specific tables are considered, with most of the results valid for the table $T_\emptyset$. Also, from the definition of the key it should be obvious that the empty set may be the key only for specific tables or tables

consisting of one string.

3 The main results

Since the keys are important in relational databases, naturally it becomes non-trivial the question of preserving keys (including the candidate keys) by signature operations of table algebras.

It is proved that the intersection preserves keys, but does not preserve the candidate keys.

Theorem 1. *Let T_1, T_2 – the tables of the schema R and K – the key of the tables T_1 and (or) T_2. Then K is the key of the table $T_1 \bigcap\limits_R T_2$. If K is the candidate key of the tables T_1 and T_2, then K can not be the candidate key of the table $T_1 \bigcap\limits_R T_2$.*

This example illustrates the second part of the Theorem 1.

Example 1. *Let* $T_1 = \begin{array}{ccc} A & B & C \\ 1 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 1 & 2 \end{array}$ *and* $T_2 = \begin{array}{ccc} A & B & C \\ 1 & 1 & 3 \\ 1 & 2 & 2 \\ 2 & 1 & 2 \end{array}$ *. Then* $K = \{A, B\}$ *is the candidate key of the tables T_1 and T_2. Then K is the key of the table* $T_1 \bigcap\limits_R T_2 = \begin{array}{ccc} A & B & C \\ 1 & 2 & 2 \\ 2 & 1 & 2 \end{array}$ *, but is not the candidate key of this table; a set of candidate keys of the table is* $\{\{A\}, \{B\}\}$.

The union do not preserve the keys, i.e., in the case where K – the key of the tables T_1 and T_2 (by schema R), then K may not be the key of the table $T_1 \bigcup\limits_R T_2$.

It is proved that the difference preserves keys, but does not preserve the candidate keys.

Theorem 2. *Let T_1, T_2 – the tables of the schema R and K – the key of the table T_1. Then K is the key of the table $T_1 \underset{R}{-} T_2$. If K is the*

candidate key of the table T_1, then K can not be the candidate key of the table $T_1 \underset{R}{-} T_2$.

This example illustrates the second part of the Theorem 2.

Example 2. *Let* $T_1 = \begin{matrix} A & B & C \\ 1 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 1 & 2 \end{matrix}$ *and* $T_2 = \begin{matrix} A & B & C \\ 1 & 1 & 3 \\ 1 & 2 & 2 \\ 2 & 1 & 2 \end{matrix}$. *Then* $K = \{A, B\}$ *is the candidate key of the tables* T_1 *and* T_2. *Then* K *is the key of the table* $T_1 \underset{R}{-} T_2 = \begin{matrix} A & B & C \\ 1 & 1 & 1 \end{matrix}$, *but not the candidate key of this table; a set of candidate keys of the table is* $\{\{A\}, \{B\}, \{C\}, \emptyset\}$.

It is proved that the selection preserves keys, but does not preserve the candidate keys.

Theorem 3. *Let K – the key of the table T. Then K is the key of the table $\sigma_P(T)$. If K is the candidate key of the table T, then K can not be the candidate key of the table $\sigma_P(T)$.*

This example illustrates the second part of the Theorem 3.

Example 3. *Let* $T_1 = \begin{matrix} A & B & C & F \\ 1 & 1 & 1 & 2 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 2 & 2 \end{matrix}$. *Then* $K = \{A, B\}$ *is the candidate key of the table T. Then K is the key of the table* $\sigma_P(T) = \begin{matrix} A & B & C & F \\ 1 & 1 & 1 & 2 \\ 2 & 1 & 2 & 2 \end{matrix}$, *where* $P(S) = true \leftrightarrow (F, 2) \in S$, *but is not the candidate key of this table; the set of candidate keys of the table is* $\{\{A\}, \{C\}\}$.

We find (in terms of active domains of attributes of tables) necessary and sufficient conditions under which non-trivial keys of unsaturated tables T and $\tilde{T}$ coincide. The requirement for unsaturation necessary to avoid cases when $\tilde{T} = T_\emptyset$, due to the fact that, as mentioned above,

the key of table $T_\emptyset$ is any subset of the attributes of the schema of this table. First, we formulate the criteria for the tables, all of whose attributes are multi-valued.

Theorem 4. *Let $R = \{A_1, \ldots, A_n\}$ – the schema of the unsaturated table T with all multi-valued attributes, and $K = \{K_1, \ldots, K_q\}$ – the non-trivial key of T. K is the key of the table $\tilde{T}$ if and only if three conditions are valid simultaneously:*

a) $\mid R - K \mid = 1$;

b) for the attribute B, belonging to the set $R - K$, the equality $\mid D_{B,T} \mid = 2$ is fulfilled;

c) for all values of $d_1 \in D_{K_1,T}, \ldots, d_q \in D_{K_q,T}$ the string $s' = \{(K_1, d_1), \ldots, (K_q, d_q)\} \in \pi_K(T)$, and there is only one such string $s \in T$, that $s' = s \mid \{K_1, \ldots, K_q\}$.

We extend the criterion of preservation of keys for tables with one-valued attributes. Let $O = \{O_1, \ldots, O_z\}$ – the set of all one-valued attributes of the table T and let $D_{O_1,T} = \{o_1\}, \ldots, D_{O_z,T} = \{o_z\}$. Let $Y = R - O$ and $T' = \begin{matrix} O_1 & \cdots & O_z \\ o_1 & \ldots & o_z \end{matrix}$. Then from the definition of the join and the active complement the equality $\tilde{T} = (\pi_Y\tilde{(}T)) \otimes T'$ implies obviously. As a result, one-valued attributes do not affect the condition (a) of Theorem 4. Thus Theorem 4 can be extended to the case of tables with one-valued attributes.

Theorem 5. *Let $R = \{A_1, \ldots, A_n\}$ – the schema of the unsaturated table T, and $K = \{K_1, \ldots, K_q\}$ – the non-trivial key of T. K is the key of the table $\tilde{T}$ if and only if three conditions are valid simultaneously:*

a) the set $R - K$ contains exactly one multi-valued attribute;

b) for the multi-valued attribute B, belonging to the set $R - K$, the equality $\mid D_{B,T} \mid = 2$ is fulfilled;

c) for all values of $d_1 \in D_{K_1,T}, \ldots, d_q \in D_{K_q,T}$ string $s' = \{(K_1, d_1), \ldots, (K_q, d_q)\} \in \pi_K(T)$, and there is only one such string $s \in T$, that $s' = s \mid \{K_1, \ldots, K_q\}$.

We find necessary and sufficient conditions under which K is a key of the table $\pi_X(T)$.

Theorem 6. *Let K – the key (the candidate key) of the table T, and $K \subseteq X$. Then K is the key (the candidate key) of the table $\pi_X(T)$.*

For researching the preservation of keys by join there were considered two cases:

1) both tables-arguments have the same key;

2) both tables-arguments have different keys.

The following theorem describes the case 1.

Theorem 7. *Let K – the key of the tables T_1 and T_2. Then K is the key of the table $T_1 \otimes T_2$. If K is the candidate key of the tables T_1 and T_2, then K can not be a candidate key of the table $T_1 \otimes T_2$.*

This example illustrates the second part of the Theorem 7.

Example 4. *Let* $T_1 = \begin{array}{cccc} A & B & C & F \\ 1 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 1 & 2 \end{array}$ *and* $T_2 = \begin{array}{ccc} B & C & G \\ 1 & 2 & 1 \\ 2 & 1 & 2 \\ 2 & 2 & 2 \end{array}$. *Then $K = \{B, C\}$ is the candidate key of the tables T_1 and T_2. Then K is the key of the table* $T_1 \otimes T_2 = \begin{array}{ccccc} A & B & C & F & G \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 2 & 2 & 1 & 2 \end{array}$, *but is not the candidate key of this table, because $\{B\}$ is the key of the table $T_1 \otimes T_2$.*

The following theorem describes the case 2.

Theorem 8. *Let K_1 – the key of the table T_1 and K_2 – the key of the table T_2. Then $K_1 \bigcup K_2$ is the key of the table $T_1 \otimes T_2$. If K_1 is the candidate key of the table T_1 and K_2 is the candidate key of the table T_2, then $K_1 \bigcup K_2$ can not be the candidate key of the table $T_1 \otimes T_2$.*

This example illustrates the second part of the Theorem 8.

Example 5.

Let $T_1 = \begin{array}{ccccc} A & B & C & F & G \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 & 1 \\ 2 & 1 & 1 & 1 & 2 \end{array}$ *and* $T_2 = \begin{array}{ccccc} B & C & F & H & J \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 & 2 \\ 1 & 2 & 2 & 1 & 2 \end{array}$. *Then $K_1 = \{A, C\}$ is the candidate key of the table T_1 and $K_2 = \{F, H\}$*

is the candidate key of the table T_2. *Then* $K_1 \bigcup K_2 = \{A, C, F, H\}$ *is the key of the table* $T_1 \otimes T_2 =$

A	B	C	F	G	H	J
1	1	1	1	1	1	1
1	1	1	1	1	2	2
2	1	1	1	2	1	1
2	1	1	1	2	2	2
1	1	2	2	1	1	2

, *but is not the candidate key of this table, because* $\{A, C, H\}$ *is the key of the table* $T_1 \otimes T_2$.

We consider the preservation of key by division only when the value of $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2$ is not empty. Since the schema of a (partial) result is the set $R_1 - R_2$, where R_1 – the schema of the table T_1, R_2 – the schema of the table T_2. We consider such cases of mutual inclusion of a key of the table T_1 and the schema R_2:

1) the key of the table T_1 does not intersect with the schema of the table T_2;

2) the key of the table T_1 intersects with the schema of the table T_2, but the key is not fully included in the schema of the table T_2.

The following theorem describes the case 1.

Theorem 9. *Let* R_1 *– the schema of the table* T_1, R_2 *– the schema of the table* T_2, $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2 \neq T_\emptyset$, K *– the key of the table* T_1 *and* $K \bigcap R_2 = \emptyset$. *Then* K *is the key of the table* $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2$. *If* K *is the candidate key of the table* T_1, *then* K *can not be the candidate key of the table* $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2$.

This example illustrates the second part of the Theorem 9.

Example 6. *Let* $T_1 =$

A	B	C	F	G
1	1	1	1	1
1	2	1	1	1
2	2	1	2	1

and $T_2 =$

F	G
1	1

. *Then* $K = \{A, B\}$ *is the candidate key of the table* T_1, K *is the key of the*

table $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2 =$ $\begin{array}{ccc} A & B & C \\ 1 & 1 & 1 \\ 1 & 2 & 1 \end{array}$ *, but is not the candidate key of this table, because* $\{B\}$ *is the key of the table* $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2$.

The following theorem describes the case 2.

Theorem 10. *Let* R_1 *– the schema of the table* T_1, R_2 *– the schema of the table* T_2, $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2 \neq T_\emptyset$, K *– the key of the table* T_1, $K \bigcap R_2 \neq \emptyset$ *and* $K' = K - (K \bigcap R_2) \neq \emptyset$. *Then* K' *is the key of the table* $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2$. *If* K' *is the candidate key of the table* T_1, *then* K' *can not be the candidate key of the table* $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2$.

This example illustrates the second part of the Theorem 10.

Example 7. *Let* $T_1 = \begin{array}{ccccc} A & B & C & F & G \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 1 & 2 \\ 2 & 2 & 1 & 1 & 1 \\ 2 & 2 & 1 & 2 & 2 \end{array}$ *and* $T_2 = \begin{array}{cc} F & G \\ 1 & 1 \\ 2 & 2 \end{array}$ *. Then* $K = \{B, C, F\}$ *is the candidate key of the table* T_1, $K' = K - (K \bigcap R_2) = \{B, C\}$ *is the key of the table* $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2 = \begin{array}{ccc} A & B & C \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{array}$ *, but is not the candidate key of this table, because* $\{B\}$ *is the key of the table* $T_1 \overset{R_1}{\underset{R_2}{\div}} T_2$.

It is proved that the rename of attributes preserves keys and candidate keys.

Theorem 11. *Let* $K = \{K_1, \ldots, K_q\}$ *– the (candidate) key of the table* T *and let* $\xi[K] = \{\xi(K_1), \ldots, \xi(K_q)\}$. *Then* $\xi[K]$ *is the (candidate) key of the table* $R T_\xi$.

4 Conclusion

We have investigated the preservation of the keys, in particular, the candidate keys, by signature operations of table algebra. It is shown that the intersection, the difference and the selection preserve the keys, but do not preserve the candidate keys (Theorem 1 - 3), and renaming of attributes preserves the keys and the candidate keys (Theorem 11); we found necessary and sufficient conditions under which the active complement preserves the keys (Theorem 4, 5). We found criterion in which the projection preserves the keys, including the candidate keys (Theorem 6). It is shown that if two tables have the same keys, then their join preserves the key and does not preserve the candidate key (Theorem 7), and if two tables have different keys, then the union of the keys will be the key of join of source tables (Theorem 8). It is proved that the division preserves the keys and does not preserve the candidate key (Theorem 9, 10). The results are of theoretical and practical interest and can be used to select the optimal keys in designing relational databases.

References

[1] E.F. Codd. *A Relational Model of Data for Large Shared Data Banks.* Communications of the ACM, vol. 13, no 6 (1970), pp. 377–387.

[2] E.F. Codd. *The Relational Model for Database Management: version 2.* Addison-Wesley, 1990.

[3] V.N. Redko, D.B. Buy. *Foundations of the theory of relational database models.* Cybernetics and Systems Analysis, vol. 32, no 4 (1996), pp. 3–12.

[4] T.M. Connolly, C.E. Begg. *Database Systems: A Practical Approach to Design, Implementation and Management. Fifth edition.* Addison-Wesley, 2009.

[5] C.J. Date. *An Introduction to Database Systems. Eighth edition.* Addison-Wesley, 2004.

[6] V.N. Redko, J.J. Brona, D.B. Buy, S.V. Polyakov. *Relational data base: table algebras and family of the SQL languages.* Academperiodica, 2001.

Aleksey Senchenko
Taras Shevchenko National University of Kyiv,
03187, Kyiv, Glushkova av., 4d.
E–mail: senchenko_as@mail.ru

Received June 27, 2015

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Human Resource Selection Process by Using Various Fuzzy Logic Techniques

Mustafa Tinkir, Burcu Doganalp, Serkan Doganalp

Abstract

Because of the fact that today human resources has been accepted as one of the most important source of competitive advantage of an organization, finding the right person for the job has become as a vital human resource management function. This paper presents mamdani and sugeno type fuzzy inference system modeling techniques being used while group decision making in the fuzzy environment and displays the methods process with an empirical application. For this purpose, as decision makers, two top managers in a business organization that is in the list of First 500 Big Industrial Organizations of Turkey have evaluated decision criteria and the candidates by using linguistic variables for the position of mechanical maintenance manager. These verbal data have been transformed into triangular fuzzy numbers for both modeling techniques. Prediction models have been obtained by using fuzzy logic and ANFIS toolboxes of MATLAB respectively and their applications on process have been realized via Simulink. All obtained prediction results have been compared with table according to prediction performances of techniques. This study shows that for deciding more accurately and effectively in human resource selection process, various fuzzy logic models are considerably suitable as an approach of fuzzy multicriteria group decision making.

Keywords: Human Resource Selection, Decision Making, Mamdani, Sugeno Fuzzy Inference System.

1 Introduction

Because of that human resources are one of the core competences for an organization to gain and enhance competitive advantage in a knowledge economy [1] today, the enterprises compete with each other for talent. In this context, finding the right person for the vacant job has become one of the most important and indispensable activities [2]. Among the functions of human resource management, human resource selection significantly affects the quality of employees and administration, and hence it has attracted intensive attention and is an important topic for the organizations [1]. Increasing competition in global markets urges organizations to put more emphasis on human resource selection process [3]. The growing importance of human resource selection process, in addition to that it is a very expensive and time taking up activity, makes the approach designing to be used in this process as a prerequisite for the organizations [2], and has brought about the usage of analytical decision making approaches [3]. The human resource selection activity can be described as a decision problem, that it is an uncertain group decision making process, and contains information which is vague and fuzzy [4]. It is clear that decision makers in charge of determining the most appropriate job candidate for the vacant position prefer to use natural language [5]. Because of expressing verbal information, using natural language causes vague information [6]. Decision making can be based on decision makers imprecise perception relying on his/her subjective ideas, experience and beliefs [7]. This situation can be considered also for human resource selection as a decision making process. It is common sense that personnel selectors tend to include as many elements as possible in their decision making process, without being able to clearly define which element has the greatest impact on the outcome of a decision [8]. Decision made under these circumstances is defined as subjective judgment [9]. Many real-world problems including human resource selection have been solved with the fuzzy logic for the last twenty five years [10]. In a vague condition, fuzzy logic approach can provide an attractive connection to represent uncertain information and can aggregate them properly [11]. Since the fuzzy logic approach

provides a simultaneous solution to a complex system of competing objectives, it seems to be a proper tool for an organization's staff allocation problem [12]. Fuzzy set theory can be applied to other business problems whenever there is a need to do modeling with imprecise reasoning processes or ambiguity in human decision-making [13]. In this context, fuzzy logic theory appears as an effective tool to incorporate imprecise judgments inherent in the human resource selection process [14]. The purpose of this study is finding the most suitable candidate for the machine maintenance manager position of the selected organization with mamdani and sugeno type fuzzy inference system modeling techniques. Triangular type membership functions are used to constitute mamdani and sugeno type-1 fuzzy inference systems.

2 Research Design and Methodology

2.1 Data Set Development

Human resource selection process for machine maintenance manager position of industrial organization has been realized in respect of Table 2. Accordingly, 5 candidates attended to the interview for the position and they were assessed in terms of decision criteria predetermined by organizations human resource management department. Interview was carried out of 16 points, and candidates' scores were calculated as a percentage. After determining interview scores, two candidates (1. candidate and 4. candidate) scored over 80 points in the interview were subjected to the English test. English test was applied and assessed out of 1200 points. As it can be seen from the Table 2, interview score of the 1. candidate was higher than the score of 4. candidate. It was because of the assessment score in respect to the business knowledge criteria. In contrast, open communication assessment score of the 4. candidate was higher than the 1. candidate. But the English test score of 4. candidate was higher. In this context, the human resource department considered that business knowledge could be improved in the course of time. Therefore 4. candidate was preferred for the proposed position.

Table 1. Human Resource Selection for Machine Maintenance Manager Position of Industrial Organization

Position	Machine Maintenance Manager					
Evaluation Criteria	Criteria Weight	1.Cndt	2.Cndt	3.Cndt	4.Cndt	5.Cndt
Open communication	4	3	2	3	4	3
Drawing lessons from mistakes	3	3	3	2	3	2
Working with strategy and targets	3	3	3	2	3	2
Skill of thinking and learning	3	3	3	2	3	2
Business knowledge	3	3	1	2	1	2
Score	16	93.75%	75%	68.75%	87.5%	68.75%
1.Candidate 4.Candidate	Out of 1200 Out of 1200	584 650	Two candidates scored over 80 points in the interview were subjected to the English test.			
Interpretation	Interview score of the 1. candidate was higher than the score of 4. candidate. It was because of the assessment score in respect to the business knowledge criteria. In contrast, open communication assessment score of the 4. candidate was higher than the 1. candidate. But the English test score of 4. candidate was higher. In this context, the human resource department considered that business knowledge could be improved in the course of time. Therefore 4. candidate was preferred for the proposed position.					

3 Fuzzy Logic Modeling Techniques

3.1 Mamdani Type Fuzzy Logic Model

The field of fuzzy system has been making a big progress motivated by the practical success in modeling and control of industrial process [15]. Fuzzy systems can be used as system modeling. In this case fuzzy modeling provides appropriate system outputs from real experimental data sets. The fuzzy logic model uses a form of quantification of imprecise information (input fuzzy sets) to generate by an inference scheme, which is based on a knowledge base of modeling. The advantage of this quantification is that the fuzzy sets can be represented by a unique linguistic expression, such as small, medium and large, etc. The linguistic representation of a fuzzy set is known as a term, and a collection of such terms defines a term-set, or library of fuzzy sets. Fuzzy logic converts a linguistic modeling strategy usually based on expert knowledge into a systems fuzzy logic modeling strategy. Fuzzy logic is made of four main components: (1) Fuzzifier; (2) Knowledge base containing fuzzy IF-THEN rules and membership functions, (3) Fuzzy reasoning; and (4) Defuzzifier interface. The basic configuration of the fuzzy system with fuzzifier and defuzzifier used in this study is shown in Figure 1.

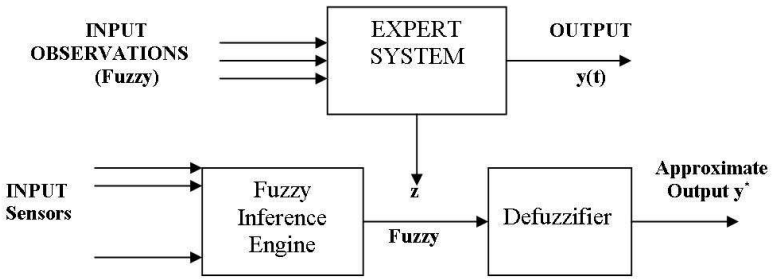


Figure 1. The basic configuration of the fuzzy system

In this paper, a Mamdani type-1 fuzzy logic modeling for human re-

source selection have been realized and compared with actual selection of selected industrial organization and results of ANFIS type fuzzy logic technique. Primarily we have obtained real data sets from organizations human resource selection process to create inputs and outputs of Mamdani type fuzzy logic model. Fuzzy logic model membership functions and rule bases have been formed by criteria of human resource selection process of the selected industrial organization.

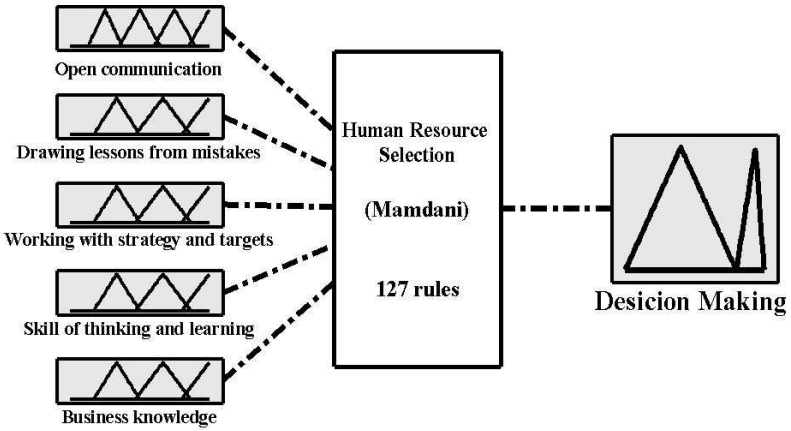


Figure 2. Fuzzy logic modeling for human resource selection

In Figure 2, fuzzy logic modeling for the selected organizations human resource selection process has been shown. In this configuration we can say that fuzzy logic model has five inputs as open communication, drawing lessons from mistakes, working with strategy and targets, skill of thinking and learning, and business knowledge. And also it has single outputs as decision making output. Moreover 127 rules used for mamdani type fuzzy inference model and membership functions for each input and output is given in Figure 3.

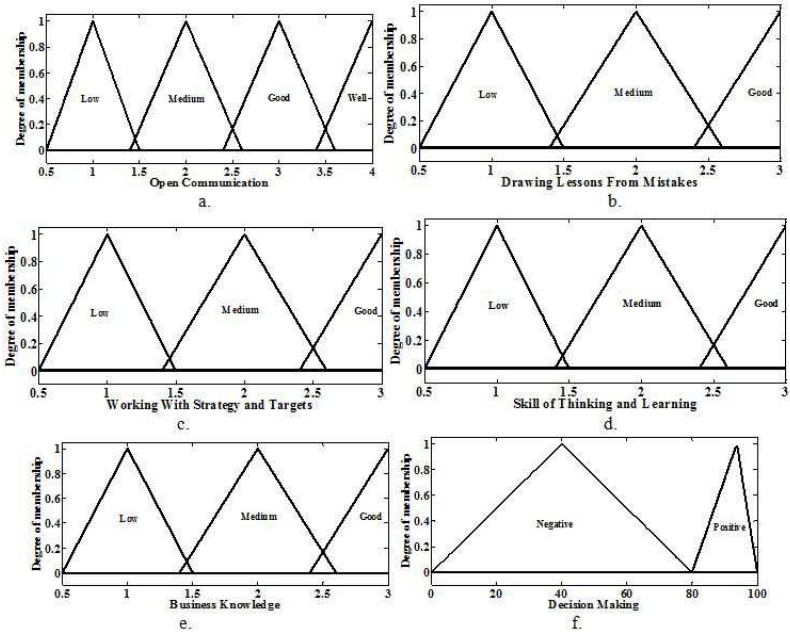


Figure 3. Membership functions of Mamdani type fuzzy logic model: *a. Membership functions of open communication input. b. Membership functions of drawing lessons from mistakes input. c. Membership functions of working with strategy and targets input. d. Membership functions of skill of thinking and learning input. e. Membership functions of business knowledge input. f. Membership functions of decision making output.*

3.2 Sugeno Type Fuzzy Logic Model

Two of the difficulties with the design of any fuzzy logic modeling are the shape of the membership functions and choice of the fuzzy rules. In fact, decision-making logic is the way in which the model output is generated. It uses the input fuzzy sets and the decision is taken according to the values of the inputs. Moreover, the knowledge base comprises knowledge of application domain and the attendant modeling goals. It consists of a database and a fuzzy logic model rule base. The fuzzification uses membership functions to determine the degree of inputs. The purpose of modeling is to obtain suitable outputs according to real human resource selection. In this study, sugeno-type inference system is used to create fuzzy logic model of proposed system. It applies a combination of the least-squares method. Fuzzy logic model of human resource selection has three membership functions for each input. Triangular type membership functions have been used in fuzzification process. Membership functions and rules of sugeno type fuzzy logic model are given in Figure 4. Fuzzy logic rule base is made of 243 rules and these rules have been determined by adaptive neural network based fuzzy inference system (ANFIS) of human resource selection. ANFIS ensures to obtain optimum range of membership functions and rules which are based on real selection data easily. But it permits only one output. Anfis outputs are constant values not fuzzy.

3.3 Adaptive Neural Network

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements. Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown in Figure 5. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are

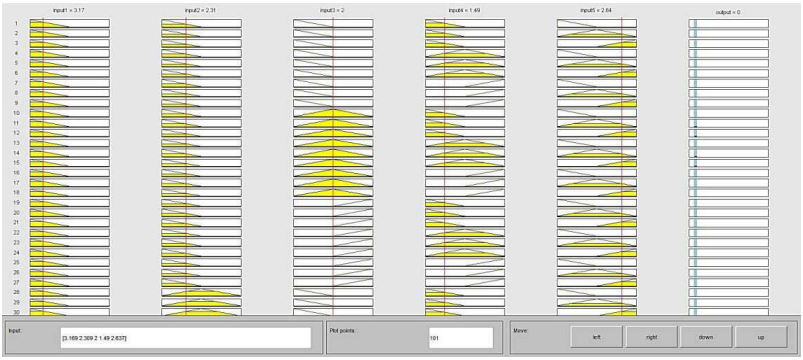


Figure 4. Membership functions and rules of Sugeno type fuzzy logic model

needed to train a network. Neural networks have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, modeling, speech, vision, and control systems.

We have followed these steps for creating ANFIS modeling shown below:

- 20 training and 5 test data have been used for neural network based on ANFIS modeling.
- The number and type of membership functions have been determined.
- Hybrid learning algorithm and 20 epochs have been chosen to train network.

3.4 Hybrid Learning Algorithm

In this study, the forward hybrid learning algorithm has been used for the neural network part of the ANFIS controllers shown in Figure 6.

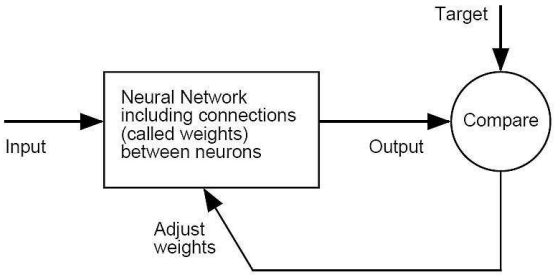


Figure 5. Neural network structure

The hybrid learning algorithm was described in the literature [16,17]. Nearly 20 epochs later, error rate is closed to 2.10-5. In the forward pass of the hybrid learning algorithm, node outputs go forward until layer 4 and the consequent are identified by the least-squares method. When the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters.

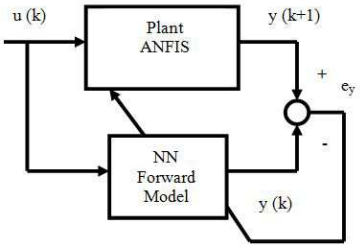


Figure 6. Training of NN forward model

The ANFIS is a fuzzy Sugeno model putting adaptive capability in framework to facilitate learning and adaptation. Such framework makes the ANFIS modeling more systematic and less reliant on expert knowledge. To present the ANFIS architecture, two fuzzy IF-THEN

rules based on a first-order Sugeno model are considered:

Rule 1: If (x is A_1) and (y is B_1), then ($f_1 = p_1x + q_1y + r_1$).

Rule 2: If (x is A_2) and (y is B_2), then ($f_2 = p_2x + q_2y + r_2$).

where x and y are the inputs, A_i and B_i are the fuzzy sets, f_i are the outputs within the fuzzy region specified by the fuzzy rule, p_i ; q_i and r_i are the design parameters that are determined during the training process [18]. The ANFIS architecture to implement these two rules is shown in Figure 7, in which a circle indicates a fixed node, whereas a square indicates an adaptive node.

When the premise parameters are not fixed, the search space becomes larger and the convergence of the training becomes slower. A hybrid algorithm combining the least-squares method and the gradient descent method is adopted to solve this problem. The hybrid algorithm is composed of a forward pass and a backward pass. The least-squares method (forward pass) is used to optimize the consequent parameters with the premise parameters fixed. Once the optimal consequent parameters are found, the backward pass starts immediately. The gradient descent method (backward pass) is used to adjust optimally the premise parameters corresponding to the fuzzy sets in the input domain. The output of the ANFIS is calculated by employing the consequent parameters found in the forward pass. The output error is used to adapt the premise parameters by means of a standard backward pass algorithm. It has been proven that the hybrid algorithm is highly efficient in training the ANFIS [19]. Fuzzy logic modeling is by far the most successful applications of the fuzzy set theory and fuzzy inference systems. Due to the adaptive capability of ANFIS, its applications to adaptive modeling and learning modeling are immediate. For this purpose, the adaptive network-based fuzzy inference system has been used to optimize the fuzzy IF-THEN rules and the membership functions to derive a more efficient fuzzy model. The proposed ANFIS model combined the neural network adaptive capabilities and the fuzzy logic qualitative approach [19].

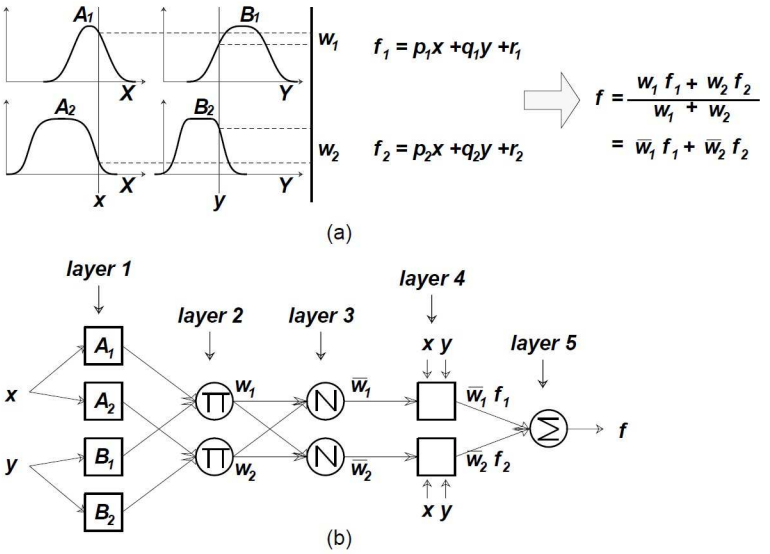


Figure 7. (a) TSK fuzzy inference system with two inputs and two rules. (b) Architecture of ANFIS of first order TSK model with two inputs.

3.5 Defuzzification Process

Once the fuzzy inference system is activated, rule evaluation is performed and all the rules are true and fired. Utilizing the true output membership functions, defuzzification is then applied to determine a crisp control action. The defuzzification is to transform the fuzzy output into an exact model output. For Sugeno-style inference, we have to choose whether wtaver (weighted average) or wtsum (weighted sum) defuzzification method to use. In defuzzification process of sugeno type fuzzy logic modeling of human resource selection, the method of weighted average (wtaver) has been used.

4 Results and Discussion

The effectiveness of the proposed modeling techniques have been tested by using MATLAB/Simulink program and the algorithm given in Figure 8 has been used to form simulation block diagram. The purpose of the models is to find the most appropriate candidate for machine maintenance manager position of the selected industrial organization according to its selection process.

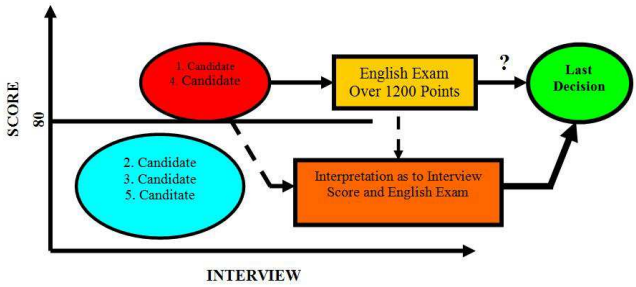


Figure 8. The algorithm used for all Fuzzy Logic Modeling Techniques

Mamdani and ANFIS type models have been created by real inter-

view results of the human resource department and five inputs have been used as decision criteria determined by the department. Last decision linguistic variable was used as single output of the models. Every candidate has been assessed via their curriculum vitae by the human resource department and then the candidates found to be appropriate after the assessment have been invited to the interview. Finally as a last stage in the selection process, two candidates whose interview scores were higher than 80 points were subjected to the English test.

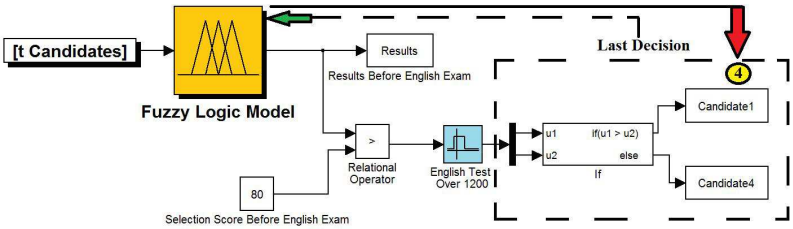


Figure 9. Simulink Diagram for Mamdani and Sugeno Type Fuzzy Logic Modeling

The program for fuzzy models has been written for this purpose by using algorithm and English exam scores of the candidates scored over 80 points in the interview have been added to simulation program to evaluate the selection process. As a result, fuzzy models have been used to select the right candidate instead of human resource departments decision making process. Comparisons of the models with real interview results are given in Table 3. According to the decision of the human resource department for the interview:

1. candidate and 4. candidate scored over 80 points in the interview were subjected to the English test. After that English test was applied and assessed out of 1200 points. Interview score of the 1. candidate was higher than the score of 4. candidate. It was because of the assessment score in respect to the business knowledge criteria. In contrast, open communication assessment score of the 4. candidate was higher than

the 1. candidate. But the English exam score of 4. candidate was higher. In this context, the human resource department considered that business knowledge could be improved in the course of time. Therefore 4. candidate was preferred for the proposed position.

Table 2. Comparison of Fuzzy Logic Techniques with Decision of Decision Makers

Interview Scores	Real Interview	Mamdani with 127 rules	ANFIS with 243 rules
1. Candidate	93.75	94.20	94.05
4. Candidate	87.50	88.01	87.92
English Test Score	Real Interview	Mamdani	ANFIS
1. Candidate	584	584	584
4. Candidate	650	650	650
Last Decision	4. Candidate	4. Candidate	4. Candidate

In addition to these selection steps, mamdani and anfis models have evaluated the 1. and 4. candidates and scored 94,20 and 94,05 points for them respectively. Other candidates were eliminated as a result of the real interview. English test scores of the two candidates were given to program to learn their scores. So convergence of the models to real selection was resulted in the last decision as 4. candidate. When the mamdani and anfis type models were compared, first interview scores of 1. and 4. candidates are close to each other. Moreover, real scores and 116 rules differences were appeared in models. Prediction results of 1. candidate are %99.5 and %99.6, results of 4. candidate are %99.4 and %99.5 for the first interview of mandani and anfis respectively. Consequently, sugeno type-1 Anfis model with a very small prediction margin is more useful for proposed selection than mamdani type fuzzy model. In mamdani model these 127 rules were written one by one according to criteria weights of the decision makers. 243 rules used in anfis model were determined automatically by anfis toolbox of the MATLAB by using mamdani data base. The adaptive capability of

the anfis model depends on its neural network base and assures future prediction about human resource selection. This capability can be tested by changing criteria weights of selection process.

5 Conclusion

This paper presents mamdani and sugeno type fuzzy inference system modeling techniques being used while group decision making in the fuzzy environment and displays the methods process with an empirical application. For this purpose, as decision makers, two top managers (human resource manager and plant manager) in a business organization that is in the list of First 500 Big Industrial Organizations of Turkey has evaluated decision criteria and the candidates by using linguistic variables for the position of mechanical maintenance manager. These verbal data have been transformed into triangular fuzzy numbers for mamdani model and also triangular type membership functions have been used to constitute both mamdani and sugeno type-1 fuzzy inference systems. According to the models, the candidates have been ranked from the best to the worst with respect to the calculated closeness coefficients. Mamdani and sugeno type fuzzy decision making models have been obtained by using fuzzy logic and ANFIS toolboxes of MATLAB software respectively and their applications on process have been realized via Simulink/MATLAB. All obtained prediction results have been compared with table according to modeling performances of used techniques. This study shows that for deciding more accurately and effectively in the human resource selection process, various fuzzy logic models are considerably suitable as an approach of fuzzy multi-criteria group decision making. Two aspects of this study that can contribute to the literature have been considered. Firstly, in the literature to date there isn't any investigation predicting the most appropriate candidate for the machine maintenance manager position by using these two methods and comparing the results of them. Therefore this study investigates the applicability of two fuzzy logic methods for predicting the aforementioned selection process and it defines which method is more useful with comparing rule tables. Secondly, expert

systems such as fuzzy logic can bring a new insight to human resource selection process which has great importance for the organizations and also affects the future performance of them. More effective decisions can be available with such fuzzy modeling techniques.

References

[1] H. Lin. *Personnel selection using analytic network process and fuzzy data envelopment analysis approaches.* Computers δ Industrial Engineering, vol. 59, (2010), pp. 937–944.

[2] P.C. Chen. *A fuzzy multiple criteria decision making model in employee recruitment.* International Journal of Computer Science and Network Society, vol. 9(7) (2009), pp. 113–117.

[3] M. Dursun, E.E. Karsak.*A fuzzy MCDM approach for personnel selection.* Expert Systems with Applications. vol. 37 (2010), pp. 4324–4330.

[4] A. Kelemenis, K. Ergazakis, D. Askounis. *Support managers selection using an extension of fuzzy TOPSIS.* Expert Systems with Applications. vol. 38 (2011), pp. 2774–2782.

[5] M.Z. Ramadan. *Effective staff selection tool: fuzzy numbers and memetic algorithm based approach.* International Journal of Engineering δ Technology vol. 9, no. 10 (2009), pp. 54–65.

[6] A.L. Zadeh. *The concept of linguistic variable and its application to approximate reasoning i.* Information Sciences vol. 8 (1975), pp. 199–249.

[7] T.L. Saaty, L.G. Vargas. *Decision making with the analytic netwok process: economic, political, social, and technological applications with benefits, opportunities, cost, and risks.* New York: Springer Science Business Media, 2006.

[8] S. Petrovic-Lazarevic. *Personnel selection fuzzy model.* International Transactions in Operational Research vol. 8 (2001), pp. 89–105.

[9] M. Ayub,Md.J. Kabir, Md.G.R. Alam.*Personnel selection method using analytic network process (ANP) and fuzzy concept.* Proceedings of 12th International Conference on Computer and Information Technology. December 21-23 2009, Dhaka, pp. 373–378.

[10] A. Golec, E. Kahya. *A fuzzy model for competency-based employee evaluation and selection.* IComputers δ Industrial Engineering vol. 52 (2007), pp. 143–161.

[11] J.R. Chang, T.H. Ho, C.H. Cheng, A.P. Chen. *Dynamic fuzzy OWA model for group multiple criteria decision making.* Soft Computing vol. 10 (2006), pp. 543–554.

[12] W. Kwak. *fuzzy set approach in audit staff planning problems.* http://www.wseas.us/e-library/conferences/jamaica2000/papers/157.pdf, Date: 21.12.2010.

[13] W. Kwak, Y. Shi, K. Jung. *Human resource allocation in a CPA firm: a fuzzy set approach.* Review of Quantitative Finance and Accounting vol. 20 (2003), pp. 277–290.

[14] E.E. Karsak. *Personnel selection using a fuzzy MCDM approach based on ideal and anti-ideal solutions.* In M. Kksalan δ S. Zionts (Eds.), Multi criteria decision making in the new millennium London: Springer-Verlag London Limited. (2001), pp. 393–402.

[15] A. Berber, M. Tinkir, S.S. Gultekin, I. Celikten. *Prediction of a diesel engine characteristics by using different modeling techniques.* International Journal of Physical Science vol. 6 no.16 (2011), pp. 3977–3990. IJPS-ISSN 1992 1950.

[16] P. Strobach. *Linear prediction theory: a mathematical basis for adaptive systems.* New York: Springer-Verlag. (1990).

[17] K.S. Narendra, K. Parthsarathy. *Identification and control of dynamical systems using neural networks.* IEEE Trans. Neural Netw. vol. 1 no.1 (1990), pp. 4–27.

[18] S.S. Haykin. *Adaptive filter theory.* (2nd Edition). New Jersey: Prentice Hall. (1991).

[19] M. Tinkir, U. Onen, M. Kalyoncu. *Modeling of neuro-fuzzy control of a flexible link. Proceedings of the Institution of Mechanical Engineers.* Part I Journal of Systems and Control Engineering. vol. 224 (2010), pp. 529–543.

Mustafa Tinkir, Burcu Doganalp, Serkan Doganalp Received June 29, 2015

Mustafa Tinkir
Institution Necmettin Erbakan University Address Department of Mechanical Engineering, Faculty of Engineering and Architecture Necmettin Erbakan University, Konya, TURKEY Phone: (+90)5394590892 E–mail: mtinkir@konya.edu.tr

Burcu Doganalp
Institution Selcuk University Address Department of Business Administration, Faculty of Economics and Administrative Sciences, Selcuk University,Konya, TURKEY E–mail: burcudoganalp@selcuk.edu.tr

Serkan Doganalp
Institution Necmettin Erbakan University Address Department of Geomatics Engineering, Faculty of Engineering and Architecture Necmettin Erbakan University, Konya, TURKEY E–mail: sdoganalp@konya.edu.tr

Part 11

Neural networks, soft computing

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Neural network based model for emotion recognition*

Veaceslav Albu

Abstract

In this paper we present the architecture of an the neural network model for recognition and classification of basic human emotions from Kinect camera input. The proposed system utilizes a learning algorithm, which combines supervised and unsupervised learning for automatic emotion classification. Using computer vision and machine learning algorithms, emotional states of multiple targets are inferred from facial expressions recorded visually by a camera. In the proposed system, human facial expressions are recorded, recognized, and analyzed to give statistical feedback of the overall emotions of a number of targets within a certain time frame. This feedback can provide important measures for user response to a chosen system.

Keywords: emotion classification, neural networks, machine learning, Kinect.

1 Introduction

Facial expressions are one of the strongest visual methods to convey emotions and one of the most powerful means for communication between humans. Understanding these emotions by the means of computer vision is one of the most impurtant tasks of modern marketing systems. Facial imaging techniques allow recording and recognition of facial expressions, passively recorded via cameras and could be applied to better measure consumer response to marketing stimuli or to detect fraudulent actions of a customer. This method allows to:

*This work was supported by NATO.NUKR.SFPP 984877. Modeling and Mitigation of Social Disasters Caused by Catastrophes and Terrorism

- get the consumer feedback without direct and intrusive questioning
- get the consistent response accross cultures
- provide a good performance on both online and offline platforms
- evaluate both video and static stimuli

Conventional methods are often unable to retrieve objective feedback from a customer and could be easily mislead by the person with fraudulent intentions. This has led to the increase in the interest in neuroscience-influenced methods, such as EEG, fMRU etc., together with the computer vision technologies.

In this paper, we provide a description of a system, inspired from both the advances in neurophysiologically-based methods and state-of-the-art machine learning algorythms.

The rest of the paper is organised as following. First, we will give a brief overview of the most prominent research in the field of emotion detection and recognition. Next, we provide a description of the proposed Kinect input algorithm and the architecture of the neural network model. In the results and discussion section, we suggest the possible application of the proposed architecture.

2 Background

Before describing the architecture of the proposed system and its applications, it is important to appreciate the academic context within the science of understanding and interpreting human facial expressions.
In our research, we utilise Damasio's definition of emotions [1], i.e. we will follow Damasio in treating emotions as guides or biases to behaviors and decision making, action plans in response to internal or external stimuli and integral part of cognition and developmental processes.

There is a number of models of emotions developed for different purposes like formalization, computation or understanding. Since this text is not a thorough review of models of emotions, we will only discuss

a couple of them. The thing that must be mentioned first of all, is the fact that all models of emotions can be classified into discrete and continuous. Discrete models work with limited sets of emotion. There might be from two (like anger and happiness) to many. Continuous models rather represent the full spectrum of human emotions in some space (usually 3D or 2D). There are combinations of those, when you bring some uncertainty into discrete models in the form of probabilities of certain emotions happening at time t and thus implement some quasi continuous model.

The original reasearch in the field of emotion recognition can be traced to Charles Darwin, upon whose work Paul Erkman et al. developted the theoretical framework for emotion detection [2]. This model was developted over years and ended up with six basic emotions: anger, disgust, fear, happiness, sadness, and surprise [2]. Important feature of Erkman's model is that those six basic emotions are semantically distinct. This approach has one minor problem, common to all discrete models of emotions. They are related to cultural and linguistic differences which do not allow those models to be truly universal. This problem is partly eliminated in the dimensional models. Dimentional models can be traced back to Wundt [3], but basic research in this field was presented by Mehrabian. He proposed to use 3D space with axes pleasure, arousal, and dominance to describe emotions. Any emotion or some blend of emotions can be represented as a point in this PAD cube. There are variation of this model, where different axes are used, but most of them end up with 3D space.

In this work, we utilise a simplified approach, which can be described as classification of emotions on 2D similarity map. The proposed network uses the input from a Kinect camera as an input and provides a classification of the detected emotion with the regard of five pre-defined emotion categories.

3 Materials and Methods

Recently, a large number of algorithms for facial expression detection were introduced [4-7]. However, the majority of them use the images

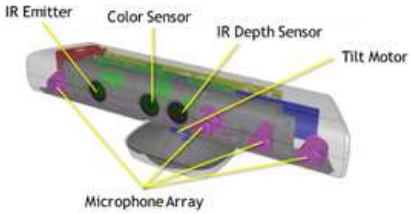


Figure 1. Microsoft Kinect sensor

obtained by web-cameras and static images. We propose a system real-time analysis of emotions (facial expression and pulse analysis) which is performed with the help of state-of-the art biometric techniques, such as Kinect data analysis (Fig.1). The resulting measurements are compared with the statistical data for distributed on-land systems (e.g. kiosks, ATMs etc.) and forecasting system will be built with the machine learning techniques.

We propose the following algorithm for visual information processing:

1 First, we use cameras and 3D sensors such as the Microsoft Kinect to detect facial features in order to recognize and classify emotions.

2 Second, we apply computer vision techniques for feature extraction and pattern recognition.

3 We apply machine learning (neural networks) for emotion detection and classification.

4 We use recorded statistical data from the machine transactions or logs. We train a modular neural network together with the emotion records to provide the analysis of the events. We can use the trained networks for further event analysis and forecasting.

4 Network architecture

The choice of neural network model was performed as following: we have examined several existing neural network architectures and selected the one with better performance results (computational costs and recognition rate) and selected the most simple network that allows the usage of the Kinect output (point cloud) within the sort time and good recognition rate.

The architecture of our model is based on the notion of the self-organized map (SOM), proposed by Kohonen [8]. This kind of neural network is trained using unsupervised learning to produce a two-dimensional map of the input space of the training samples. SOM detects regularities and correlations in its input and adapt their future responses to that input. The neurons of competitive networks learn to recognize groups of similar input vectors in such a way that neurons, which are located physically near each other in the neuron layer, respond to similar input vectors.

The main idea in the SOM learning process is that for each input vector the winner unit is selected. The winner unit (which is called best matching unit, BMU) and the nodes in its neighbourhood are changed closer to in the input data. If the number of available inputs is restricted, they are presented re-iteratively to the SOM algorithm. The Kohonen′ network is trained with the method of successive approximations.

The conventional SOM algorithm has a number of restrictions, and the main one is its ability to deal only with the vectorized data. To solve this problem, a number of modifications of the conventional SOM have been proposed. We used one of these modifications as a basis for constructing our model.

Tokunaga and Furukawa have proposed a significant variation of the conventional SOM, called the modular network SOM (mnSOM) [9]. In their model, each vector unit of the conventional SOM is re-

placed by a functional module. These modules are arrayed on a lattice that represents the coordinates of the map. Authors regard the case of a multi-layer perceptron (MLP) module as the most commonly used type of neural network. This architecture was designed to keep the backbone algorithm of the SOM untouched. The algorithm of the mnSOM is a generalization of a conventional SOM that inherits many properties from a conventional version of the algorithm and also adds several new original properties.

This architecture has the number of advantages. First, every module in the mnSOM has the capability of information processing and can form a dynamic map that consists of an assembly of functional modules. Second, the mnSOM combines supervised and unsupervised learning algorithms: at the MLP-level, the network is trained by a supervised learning algorithm, i.e., the back propagation at the MLP module level, while the upper SOM level is described in an unsupervised manner.

For the purposes of this study, we substituted the multilayer perceptrons with RBF network modules.The usage of RBFs instead of the MLPs adds the number of advantages,such as avoiding a problem of local minima, while preserving the ability of the NN to form a dynamic map. A radial basis function neural network (RBFN) is an artificial neural network that uses radial basis functions as activation functions. The typical RBFN architecture consists of three layers: an input layer, a hidden layer of J basis functions, and an output layer of linear output units. The activation values of the hidden units are calculated as the closeness of the input vector x_i to an I-dimensional parameter vector f_i^j associated with hidden unit u_j.

In a RBF network there are three types of parameters that need to be chosen to adapt the network for a particular task: the center vectors u, the output weights w_j, and the RBF width parameters σ.
The variety of the training algorithms for RBFNs exists. One of the possible training algorithm is gradient descent. In gradient descent

training, the weights are adjusted at each time step by moving them in a direction opposite from the gradient of the objective function (thus allowing the minimum of the objective function to be found).

The model of the main module of the proposed network represents a modification of the conventional SOM, where each vector unit of the conventional SOM is replaced by a functional RBF module. These modules are arrayed in a lattice that represents the coordinates of the feature map.

The architecture of the SOM of RBFs module has a hierarchical structure: it consists of two levels, which we will call the RBF-level and the SOM-level of the network. At the first level, the architecture of our network represents k RBF- networks, which are the modifications of the Poggio and Edelman network. Since each module represents a certain "functional feature" determined by the model architecture, the SOM-level the SOM of RBFs represents a map of those features.

The proposed network solves an approximation problem in a high-dimensional space. Recognizing of an object is equivalent to finding a hyper-plane in this space that provides the best fitting to a set of training data. The training data represents a vector with coordinates of 2D projections of 3D objects, taken at each degree of rotation.

The architecture of the SOM of RBFs module has a hierarchical structure: it consists of two levels, which we will call the RBF-level and the SOM-level of the network. At the first level, the architecture of our network represents k RBF- networks, which are the modifications of the Poggio and Edelman network.

The proposed network solves an approximation problem in a high-dimensional space. Recognising of an object is considered to be equivalent to finding a hyper-plane in this space that provides the best fitting to a set of training data. The training data represents a vector with coordinates of 2D projections of 3D objects, taken at each degree of

rotation.

Let x_i denote the units of the input vector, o define the output of each RBF-module, u_j^k define the RBF centres, σ_j^k define the variance, w_j^k define weights, and n to be the number of hidden units, where j defines the j^{th} hidden unit and k defines the k^{th} RBF-module. Then the conventional SOM algorithm can be rewritten as follows.
In the first step, the weights w_j^k are defined randomly in the interval [0 0.5]. In the evaluative process, we calculate all outputs for all of the inputs in single RBF-unit according to the following rule:

$$o(x) = \sum_{j=1}^{n} w_j exp\{\frac{-(x-u_j)^2}{{\sigma_j}^2}\} \tag{1}$$

This calculation process is repeated for all of the RBF-units using the same input x. After evaluation of all of the outputs for all inputs, the errors for all the datasets are calculated:

$$E_i^k = \frac{1}{2}(y - \frac{1}{1+e^{-o(x)}})^2 \tag{2}$$

where y defines the desired output. E_i^k defines the error for the i^{th} dataset for the k^{th} module. The desired output equals 1 for all the RBF modules.

In the competitive process, the module that minimizes the error is determined as the winner module.

In the cooperative process, the learning weights are calculated using the neighbourhood function α, which decreases with the calculation time.

$$\alpha(r_i) = \frac{e^{\frac{-(r_i - r_v)}{2\xi_j^2}}}{\sum_{i-1}^{n} e^{\frac{-(r_i - r_v)}{2\xi_j^2}}} \tag{3}$$

where r_i denotes the position of the i^{th} RBF-unit in the map space, r_v expresses the position of the module with the minimal error and ξ

is the parameter of the neighbourhood function. The neighbourhood function area is decreased monotonically each epoch of learning.

In the adaptive process, all of the modules are updated by the back-propagation learning algorithm:

$$\Delta w_j^k = \eta \partial E_i^k / \partial w_j^k (t-1) \tag{4}$$

and

$$w_j^k(t) = w_j^k(t-1) + \Delta w_j^k \alpha(r_i) \tag{5}$$

so (4) can be rewritten as follows

$$\Delta w_j^k = \eta(y - \frac{1}{1+e^{-o(x)}})(y - \frac{1}{(1+e^{-o(x))^2}})e^{\frac{-(x-u_j)^2}{2\sigma_j^2}} \tag{6}$$

The centres of the RBF-units are updated according to the following rules

$$\Delta u_j^k = \eta \partial E_i^k / \partial u_j^k (t-1) \tag{7}$$

and

$$u_j^k(t) = u_j^k(t-1) + \Delta u_j^k \alpha(r_i) \tag{8}$$

i.e.,

$$\Delta u_j^k = \eta(y - \frac{1}{1+e^{-o(x)}})(y - \frac{1}{(1+e^{-o(x))^2}})w_j \frac{x-u_j}{2\sigma_j^2} e^{\frac{-(x-u_j)^2}{2\sigma_j^2}} \tag{9}$$

The learning is repeated until all of the modules are updated. Training continues until the network reaches a steady state.

5 Results and Discussion

In this work, we propose the NN architecture for emotion classification. The input of the model is the real-time data, provided with the Kinect camera (Fig.2).

The network output represents the activation map, the activation of each module shows the belonging of the detected expression to one

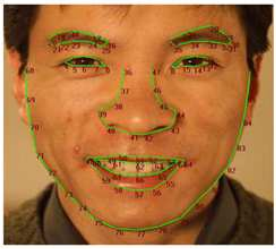


Figure 2. The input of the RBF-SOM network: 87 key facial features, captured by Kinect Microsoft Software.

of five basic emotions. For the purposes of this study, we selected five basic emotions, which are locates on a square plane, divided into 25 parts. The winning module represent the most plausible emotion. This approach allows defining the most plausible emotion or emotions (since the most active module can be defined between two emotions). In this paper, we used only five emotions, but the usage of a larger number of emotion labels is also possible (Fig.3).

We described the properties of the proposed neural architecture for hierarchical visual perceptual processing, composed of modules resembling human visual system.By introducing this architecture, our model appeared to be capable of performing recognition and classification of simple emotions and creating a similarity map of these emotions.

From the point of view of emotion recognition system, current approach is close to the one proposed by Paul Ekman [2], but differs in the type of machine learning techniques, equipment and the number of emotions (eh utilises six basic emotions: anger, disgust, fear, happiness, sadness, and surprise, while we use only five expressions: sad, angry, neutral, happy, engaged).

The research, described in this work, constitutes the tiny part of the

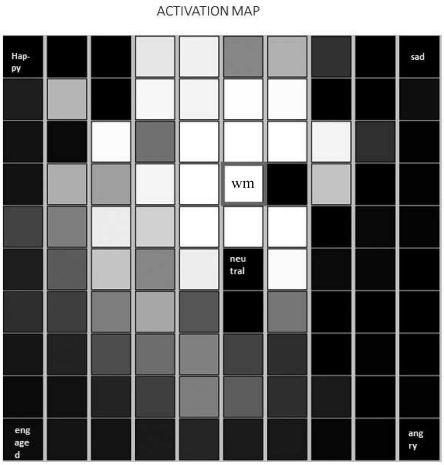


Figure 3. The output of the RBF-SOM network: the activation map describes the performance of the neural network architecture: the networks activated one of five basic emotions (black color refers to absence of activation and the white color refers to active module). In this case, the most active or winning module (marked with red square) is closer to "neutral" expression.

spacious area of visual object recognition. It includes the description of main research in the domain of emotion recognition with the means of computer vision system with the emphasis on the neural network architectures. It could be further extended in order to add the number of emotions and to implement different machine learning techniques. An application example of this research is a camera system embedded in a machine that is used frequently, such as an ATM or information kiosk.

References

[1] A. Damasio. *Descartes' Error: Emotion, Reason, and the Human Brain.* Putnam Publishing, (1994).

[2] P. Erkman. *Handbook of Cognition and Emotion.* New York, NY: John Wiley and Sons Ltd. (1999).

[3] W.M. Wundt. *Classics in the history of psychology.* .

[4] J.F. Cohn *Foundations of human computing: facial expression and emotion*,ICMI 2006: Proceedings of the 8th international conference on Multimodal interfaces, ACM, New York, NY, USA. (2006), pp. 233–238.

[5] Y.-L. Tian, T. Kanade, J. Cohn. *Facial expression analysis.* S. L.

[6] M. Meulders, P. D. Boeck, I. V. Mechelen, A. Gelman. *Probabilistic feature analysis of facial perception of emotions.* Journal Of The Royal Statistical Society Series C 54. (2005), pp. 781–793.

[7] B. Jiang, M. F. Valstar, M. Pantic. *Facial Action Detection using Block-based Pyramid Appearance Descriptors.* The proceedings of the ASE/IEEE International Conference on Social Computing (SocialCom 2012).

[8] T. Kohonen. *Self-organizing maps.* Berlin: Springer-Verlag. (2001).

[9] K.Tokunaga, T. Furukawa. *Modular network SOM.* Neural Networks (2009), v.22, pp. 82–90.

Veaceslav Albu

Received July 17, 2015

Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
5 Academiei str., Chişinău, MD-2028, Moldova
E–mails: vaalbu@gmail.com

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Ambient intelligence in decision support systems

Alexei Averkin

Abstract

The paper describes the basics of creation of a new generation of intelligent systems – ambient intelligence. We propose the concept of the development of intellectual environment as cognitive-regulatory meta-agent with the use of intelligent technology of decision making, getting a considerable amount of information from the sensors arranged in the form of wireless sensor networks.

Keywords: ambient intelligence, wireless sensor networks, cognitive agents

1 Introduction

The aim of the paper is to describe the new generation synergistic of intelligent systems – ambient intelligence, as well as the development of a new concept of the intellectual environment as cognitive-regulatory meta-agent. We consider stages of the evolution of intelligent systems, which led to appearance of the agent-oriented approaches, as well as the main features of cognitive agents, and the key technologies of formation of ambient intelligence. The basic architecture of ambient intelligence is shown. We introduce the concept of ambient intelligence as artificial agent, the ongoing process of cognitive-regulatory coordination with the use of intelligent technologies, including natural language dialogue with the people. We build the ontology of wireless sensor networks as principal means in obtaining and processing information via ambient intelligence.

Future prospects in research are related to the development of logical-linguistic model of dialogue between the sensors of the sensor network and the implementation of procedures of data mining, sensor mining and knowledge discovery.

This work was carried out with the financial support of the Russian Foundation for Basic Research, projects 14-07-00653 and 13-07-00893.

2 Evolution of Intelligent Systems

Stages of the development of intelligent systems can be displayed using the chain of "heuristic search – simple expert systems – hybrid intelligent systems – intelligent network and multi-agent systems – intelligent environment". This chain reflects the leading trends of modern AI: the integration of diverse technologies, distribution, agent orientation, anthropocentrism, in particular, modeling NON-factors, granulation of information and transition to an anthropomorphic (granular) calculations [1-5].

Creation of the integrated intelligent systems is not just a union, but a mutual adaptation and joint evolution of heterogeneous components. Integration acts as a necessary condition for hybridization [6]. Hybrid systems in AI consist of two or more dissimilar integrated subsystems united under a general purpose or joint action (although these subsystems can be of different nature and different description languages). They use two or more different computer technologies. In particular, the association of expert systems with databases and application packages has led to the emergence of hybrid expert systems.

Soft computing [6,7] have become another important direction in the development of hybrid intelligent systems with NON-factors [3,4]. In this methodology, three aspects of intelligent behavior – fuzzy information processing, learning and adaptation in the evolutionary process – are bound together, providing fuzzy production models in training of the neural network.

Further development of the integrated and hybrid intelligent systems leads to the formation of synergistic intelligent systems, i.e. complex, self-organizing, evolving systems, consisting of coherent and co-

operating components. Creation of such systems requires the development of synergetic methodology in artificial intelligence [1]. Specific examples of synergistic intelligent systems are multi-agent systems of different classes.

Agent is an open, active and goal-oriented system, which is capable of forming its own behavior in fully defined environment. We can select classes of natural and artificial, physical and virtual, reactive and intelligent agents [1]. If artificial intelligent agents are endowed with their own motivations and are capable of forming their own goal (goal-oriented agents), then they are called intentional. Otherwise, when the artificial intelligent agents obtain goals from natural targeting agents, they are often called reflex agents (which understand and realize the goals and interests of users).

In addition, artificial intelligent agents are divided into cognitive, deliberative and communicative. In the case of purely communicative agents, inner world model is converted mainly to model communication patterns, consisting of participants, process of communication and desired result. Deliberative (reasoning) agents are able to carry out quite complex arguments of different types (for example, abductive, deductive, by analogy) and to make decisions on their basis or perform actions, changing environment.

Artificial cognitive agents have advanced knowledge of the subsystem, which provides the construction of internal models of the external environment, in particular, the models of other agents, as well as models of their own state.

3 Cognitive agents

The basic cognitive processes are processes of external world perception and its generalized presentation, understanding of the laws of interaction, behavior and training. They include resource allocation process (in particular, attention), forecasting and planning behavior, formation of their own arguments about their own states and states of other objects and agents.

Key features of cognitive processes that should be considered in

the development of cognitive agents are the following: 1) cognitive process is an open system, based on existing knowledge and perception of current data; 2) cognitive process generates hypotheses, rather than conclusions; these hypotheses require confirmation or refutation; 3) cognitive process of the environment should not be separated from the organization the agent's actions (as an information process, local changes in the environment or physical movement). Thus, the system of artificial cognitive agent should monitor the environment and get information from sensor subsystem.

Thus, the intelligent agent is the chain "data-information-knowledge-meta-knowledge", i.e. support of baseline data, recycling information, forming opinions, knowledge and plans for the construction of knowledge about knowledge.

The implementation of this chain suggests formation of an artificial cognitive agent with the following mechanisms of understanding: a) links between objects or events, b) samples and examples of regulatory or situational behavior; c) general situation.

At the same time, artificial cognitive agents should be able to communicate (to dialogue) with user in a limited natural language. Under the dialog we understand a sequence of communicative acts between a human and ICA, including the ability to change the role ("active-passive" participant or "speaker-listener" in the process of communication). Any dialogue involves the exchange of messages related to the change in objectives and agent's state. Dialogue with artificial human agent includes both targeting and instructions, transmitted by human agent, and feedback (when a person tells an agent to clarify the original instructions), as well as information on the current situation or information on achievement of the goal [8]. Dialogue can beorganized in a variety of ways, in particular, by means of natural language (text or voice).

In general, the status of a cognitive agent is determined by acquisition, integration and use of diverse information from various sources, including: 1) human user; 2) own database/agent's knowledge; 3) sensors [8].

4 Ambient Intelligence

The concept of "ambient intelligence" ("intellectual space", "intellectual environment"), entered into use due to the specialists from Phillips in 1998, is the enhanced physical environment, which incorporates technical devices, sensitive to the presence of people and responsive to this presence. This implies recognition of users sensitivity, understanding of their preferences and context, forecasting the behavior.

The term comes from the English words combinations Ambient Intelligence and Smart Environments, and is a generic term Ambient Intelligence (AmI). Under the Smart Environments we understand, first of all, technical devices (sensors, actuators and computer network) that support the establishment and functioning of this environment [9-11].

Ambient intelligence is a promising interdisciplinary information and communication technology, generated at the intersection of cybernetics, artificial intelligence, computer science, cognitive science, activity theory, mechatronics, and ergonomics. It should be noted that we are talking about a new generation of "human-machines environment", which is usually studied in ergonomics.

Ambient intelligence (in a broad sense) is a new concept for the implementation of interactions "human-machines environment", when people are surrounded by intelligent and intuitive interfaces and built-in objects of everyday life.

Integrated technology of intellectual environment should be transparent, built in the surrounding reality, well-adapted, providing a simple and convenient interaction "human-machines" and "human-technique-human" function if there is the need [9].

Ambient intelligence, the populations of various technical devices (sensors, actuators), and "wired" network, which connects all of these devices ("Internet of Things") should naturally serve to support professional activities of people through the use of intelligent technologies. By decreasing the amount of such artificial systems and devices, establishing close relationships between them, and increasing their integration into the real physical environment via the procedures like "dissolve" in this environment, we help the users to perceive only user-friendly

interface with the environment. Thus, ambient intelligence (in the narrow sense) should be considered as a new round of development of information technology synergies on the basis of artificial intelligence.

In this paper we propose the concept of ambient intelligence as cognitive-regulatory meta-agent, which basic architecture is expressed in the following form: ambient intelligence = a distributed system of perception of physical and technical parameters of the environment + intelligent core + intelligent distributed system with impact on the physical and technical environment. Cognitive-regulatory coordination of meta-agent (when it interacts with the physical and technical environment) is characterized by a pair of "cognitive interval – level of regulation". Thus, the artificial meta-agent, which interacts with the physical and technical environment to ensure comfortable conditions for the activity of natural agents, as well as to monitor technical objects and support management decision-making, includes the following main components:

1) tools of organizing and processing knowledge of reasoning, action planning (ontology, logic, knowledge base, tools of data mining and machine learning, natural language processing tools, intelligent decision support systems, etc.);

2) artificial sensory systems (primary sensors and sensor networks, vision systems, voice recognition systems, radio frequency automatic identification RFID, satellite navigation system GPS or GLONASS, etc.);

3) artificial tools of implementation impacts on the environment (effectors, drives, mechatronic devices).

5 Distributed Sensor Systems: Sensor Networks

A crucial component of the artificial intelligent system environments are sensors, forming a sensor network. Sensor network is a set of heterogeneous, distributed sensors, covering a large area, which interact with each other in order to extract and aggregate information from in-

dividual, local, and "raw" data. The network is seen as a complement to traditional technologies for collecting, processing and transmitting information, and is usually limited to static sensors that receive and process flows of homogeneous data.

In recent years, the creation of distributed systems of technical perception using wireless sensor networks (WSN) [12] is one of the priority directions in the development of information systems and intelligent technologies. WSN is a flexible, autonomous, self-organizing network, which includes a variety of different heterogeneous sensors and actuators, connected to each other by radio. The coverage area of such network can range from few meters to tens of kilometers (by the ability of relaying messages from one cell to another).

Wireless sensor networks have a number of important advantages:

- Comparative cheapness (in accordance with the "core network law", the value of the sensor network is reduced by increasing the number of its elements);
- Rapid installation of equipment at the facility;
- High survivability of the system (in case of failure of individual elements the system continues to operate and provides the user with the necessary information);
- Mobile technologies (having finished the kernel, you need to make only minimal changes in software).

Each node of the sensor network may comprise various sensors to monitor external environment, microcomputer and wireless communication module. This allows the device to measure, independently carry out primary data for processing and communication with external information systems. The exchange of information between the nodes of the system takes place through the ZigBee wireless protocol. This protocol provides the possibility of implementing a wireless communication with low power consumption for a variety of applications that perform the functions of monitoring and/or control.

To realize the ambient intelligence paradigm the embedded soft computing approach in wireless sensor networks was suggested. This approach means a combination of embedded fuzzy logic and neural networks models for information processing in complex environment with uncertain, imprecise, fuzzy measuring data. It is a generalization of soft computing concept for the embedded, distributed, adaptive hybrid systems of decision-making [13, 14]

The main part of our embedded soft computing and soft computing approaches is Smart Node (SN) model for WSN. The core of SN is Fuzzy Engine that consists of three modules: knowledge base (a set of fuzzy production rules), fuzzification and defuzzification modules (which transform numerical measurements in linguistic form and vice-versa). The output of SN can approximate any function of input parameters, e.g. when it is impossible or difficult to measure a parameter, it can be computed by SN with the use of special rules from knowledge base. Similarly, special rules can be created for data fusion, clusterization, aggregation, routing and power consumption. The knowledge base of SN can be created as a result of knowledge acquisition from expert or by supervised neural network learning. Utilization of knowledge in nodes can significantly improve the resource and energy efficiency, i.e. by application-specific data caching and aggregation in an intermediate node.

References

[1] V. B. Tarasov. *From multi-agent systems to intellectual organizations: philosophy, psychology, computer science.* – Moscow: Editorial URSS, 2002. (in Russian).

[2] L. A. Zadeh. *Toward a Theory of Fuzzy Information Granulation and its Centrality in Human Reasoning and Fuzzy Logic.* Fuzzy Sets and Systems. – 1997. – Vol.90, pp. 111–127.

[3] A. S. Narinyani. *NON-factors and knowledge engineering: from the naive to the formalization of natural pragmatics.* Proceedings

of the IV-th National Conference on Artificial Intelligence (CAI-94, Rybinsk, September 1994). V.1. – Tver: FIA, 1994. – pp. 9–18. (in Russian).

[4] *Artificial News Intelligence.* – 2004. – N2. (in Russian).

[5] A. Bargiela, W. Pedrycz. *Granular Computing: an Introduction.* – Dordrecht: Kluwer Academic Publishers, 2003.

[6] *Fuzzy hybrid systems. Theory and Practice.* / Ed. N. G. Yarushkinoy. – Moscow: FIZMATLIT, 2007.

[7] L. A. Zadeh. *Fuzzy Logic, Neural Network and Soft Computing.* Communications of the ACM. – 1994. – Vol.37, N3. – pp. 77–84.

[8] V.B. Tarasov, A.P.Kalutsky, M.V. Svyatkina. *Granular, fuzzy and linguistic ontology to ensure mutual understanding between cognitive agents.* Open semantic technologies of intelligent systems. Materials II-nd International Scientific and Technical Conference (Minsk, BSUIR, 16-18 February 2012). – Minsk: BSUIR, 2012. – pp. 267–278. (in Russian).

[9] E. Aarts, R. Harwig, M. Schuurmans. *Ambient Intelligenc. The Invisible Future: The Seamless Integration of Technology into Everyday Life* / Ed. by P. J. Denning. – New York: McGraw-Hill Companies, 2001.

[10] *Handbook of Ambient Intelligence and Smart Environments.* Ed. by H. Nakashima, H. Adhajan and J. C. Augusto. – New York: Springer Verlag, 2010.

[11] *Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives.* / Ed. by N.-Y. Chong and F. Mastrogiovanni. – New York: IGI Global, 2011.

[12] W. Dargie, Ch. Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice.* – New York: John Wiley and Sons, 2010.

[13] A. N. Averkin, O. P. Kuznetsov, A. A. Kulinich, N. V. Titova. *Decision-making support in weakly structured subject domains: Analysis of situations and evaluation of alternatives.* Journal of Computer and Systems Sciences International, Volume 45, Issue 3, pp. 469–479.

[14] A. N. Averkin, A. G. Belenki. *Soft Computing in Wireless Sensors Networks.* In EUSFLAT Conf. (1)(2007), pp. 387–390.

Alexei Averkin

Received July 29, 2015

Institute of Computer Science,
Russian Academy of Sciences
E–mail: averkin2003@inbox.ru

Proceedings of the Workshop on Foundations of Informatics
FOI-2015, August 24-29, 2015, Chisinau, Republic of Moldova

Petri nets to model disaster prevention*

Inga Titchiev

Abstract

The aim of this article is to form preliminary concepts of simulation system by means of Petri nets for independent disaster prevention organizations as a tool to heighten consciousness of potential damage used to simulate evacuation behavior of inhabitants in a case of disaster.

Keywords: Petri nets, disaster, modeling.

1 Introduction

When a social disaster [3, 4] happens, it can lead to other accidents and catastrophes and it may be necessary to keep human health and in some cases, human life. The best method for protecting human life is to smoothly evacuate inhabitants to safe places until danger has passed. In order to determine important features related to the successful evacuation of people, it is proposed using of formal method like Petri Net[1, 2]. Petri-net is a mathematical model for expressing a sequentially, asynchronous and parallel system. They have intuitive graphical representation and high power of modeling.

A Petri Net is a bipartite graph with two types of nodes: places and transitions interconnected by arcs, which connect only different types of nodes.

In Petri nets, the process dimension specifies which actions must be performed and in what order. For modeling people flows by means of Petri Nets, the transition will be done directly: actions will be modeled by transitions, the place where the people are will be modeled by places, people will be modeled by tokens and dependencies by arcs.

* This work was supported by NATO.NUKR.SFPP project Ref. Nr. 984877

2 Petri Nets. Definition.

As it was mentioned above, Petri Net is constituted by places, arcs, transitions and tokens. If all conditions are fulfilled (in places), transition will work (it is called "firing"), token is eliminated from input place, and is added to output place. Dynamic action is expressed when token moves between input place and output place. Formally we have the following definition and rule:

Definition 2.1 The 4-uple PN=(P, T, F, W) is a *Petri Net*, where P is a set of places, T is a set of transitions, F is a set of arcs, W is a weight of arcs.

The firing of a transition is possible according by the *firing rule*:

1. *Applicability rule*

   A transition can fire if in all its input places there are at least as many tokens as is the weight of corresponding arcs.

2. *Calculation rule*

   If a transition is possible, after firing it removes tokens from input places and adds them to output places.

2.1 Modeling using Petri Nets

Suppose that there is a two-floors building as it is specified in Figure 1. M1-M12 are rooms, M11-M121 are the doors. Petri Net of this building, the first floor is given in Figure 2.

Rooms and doors are modeled using places M1-M81, movements from the rooms to the doors or from the doors into the rooms are modeled by transitions t1-t14. Each inhabitant is modeled by one token, respectively. The people are accumulated in the places. The transfer function is used, which takes into account the time spent in the queue (moving time of a human in the room) and the density and flow rate.

In its turn, the flow rate in every room depends on the flux density on it. Flow rate at the rooms following after the first one, it depends on the flow rate, which is expressed in dependence of the intensity of

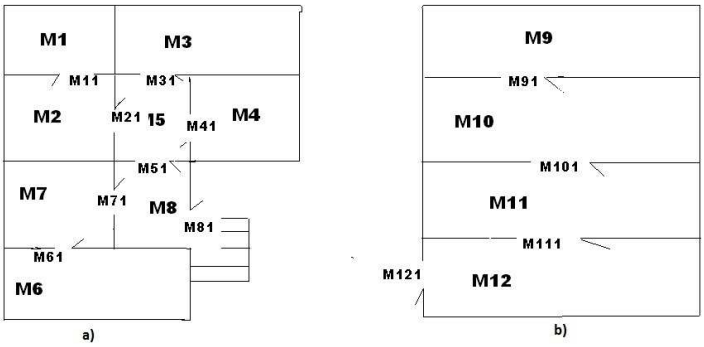


Figure 1. Building plan

flux in the previous rooms. Time of evacuation is directly proportional to the length of the way and inversely proportional to the flow rate of the people in this way.

For the simulation of evacuation time of people from buildings it is necessary to do the following steps:

1. Prepare a building plan.
2. Translate building plan in terms of Petri nets.
3. Set the transition conditions according to a mathematical model.
4. Perform initial marking (define vectors of probability distribution of having people in rooms).
5. Run the simulation program for calculating the estimated time of evacuation.

The final goal of the Petri net is the automatic analysis of the properties of modeling system, such as boundedness, liveness, deadlock, safeness.

Petri net, used for modeling of people evacuation, must fulfill the following properties:

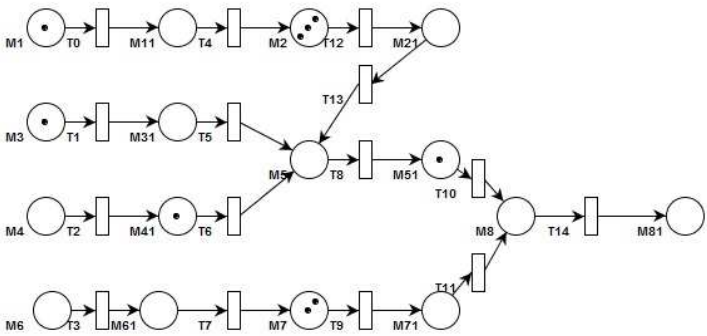


Figure 2. Petri net representing first flor a)

1. it must be bounded (a finite set of states, leading to a finite number of steps necessary for evacuation).

2. it must be safe (transitions do not influence each other, each door and room works independently of one another).

3. it must be conservative (number of people is constant, do not appear new people, they are accumulated in the last place).

4. it must be without deadlock (dead transitions do not occur, it means that persons who are unable to evacuate must not appear).

3 Conclusions

In this study, a method of Petri-net was proposed for simulation system that represents emergency evacuation of people in case of social disaster. This method allows checking such properties as boundedness, liveness, deadlock, safeness.

References

[1] O.Pastravanu. *Aplicatii ale retelelor Petri in studierea sistemelor cu evenimente discrete.* Ed. Gh. Asachi, Iasi, 2002, 238 p.

[2] J. L. Peterson. *Petri Net Theory and The Modeling of Systems*, Prentice Hall, 1981.

[3] Takashi Minamoto, Yoshifumi Nariyuki, Yasuhiro Fujiwara, Atsushi Mikami. *Development of Tsunami refuge PETRI-NET simulation system utilizable in independence disaster prevention organization*, The 14^th World Conference on Earthquake Engineering October 12-17, 2008, Beijing, China.

[4] O.Tsujihara, K. Terada, T. Sawada. *Development of simulation system of spreading fire occurring simultaneously in many places in an earthquake using Petri-net.* Journal of Applied Computing in Civil Engineering 14:11, 2005, pp. 129–136.

Inga Titchiev, Received July 12, 2015

Inga Titchiev
Institute of Mathematics and Computer Science
Address 5, Academiei street, Chisinau, Republic of Moldova
Phone: 0 22 73 81 30
E–mail: inga.titchiev@gmail.com

Table of contents

Part 1. Theory of computing

Part 2. Theoretical aspects of software system development

Part 3. Natural computing

Part 4. Theoretical issues in automated reasoning

Part 5. Logics in informatics

Part 6. Formal languages and automata

Part 7. Semantic technologies

Part 8. Natural language processing

Part 9. Cryptography and security

Part 10. Databases, artificial intelligence

Part 11. Neural networks, soft computing