Computer Science Journal of Moldova, vol.**23**, no.3(69), 2015

Convex graph covers

Radu Buzatu, Sergiu Cataranciuc

Abstract

We study some properties of minimum convex covers and minimum convex partitions of simple graphs. We establish existence of graphs with fixed number of minimum convex covers and minimum convex partitions. It is known that convex p-cover problem is NP-complete for $p \geq 3$ [5]. We prove that this problem is NP-complete in the case $p = 2$. Also, we study covers and partitions of graphs when respective sets are nontrivial convex.

Keywords: Convexity, graphs, convex covers, convex partitions.

1 Introduction

We denote by $G = (X; U)$ a simple graph with vertex set X and edge set U. The set of all vertices adjacent to $x \in X$ in G is denoted by $\Gamma(x)$.

Now we remind some notions defined in [1]: a) *metric segment* $\langle x, y\rangle$ is the set of all vertices lying on a shortest path between vertices $x, y \in X$; b) a set $S \subseteq X$ is called *convex* if $\langle x, y\rangle \subseteq S$ for all $x, y \in S$; c) *convex hull* of $S \subseteq X$, denoted $d - conv(S)$, is the smallest convex set containing S.

A family of sets is called *convex cover* of $G = (X; U)$ and is denoted by $\boldsymbol{\mathcal{P}}(G)$ if the following conditions hold:

1) every set of $\boldsymbol{\mathcal{P}}(G)$ is convex in G;
2) $X = \bigcup_{Y \in \boldsymbol{\mathcal{P}}(G)} Y$;
3) $Y \not\subseteq \bigcup_{\substack{Z \in \boldsymbol{\mathcal{P}}(G) \\ Z \neq Y}} Z$ for every $Y \in \boldsymbol{\mathcal{P}}(G)$.

If $|\boldsymbol{\mathcal{P}}(G)| = p$, then this family is called *convex p-cover* of G and is denoted by $\boldsymbol{\mathcal{P}}_p(G)$. The concept of *convex p-cover* of a graph was defined in [5]. In particular, a family $\boldsymbol{\mathcal{P}}(G)$ is called *convex partition* of graph G if it is a convex cover of G and any two sets of $\boldsymbol{\mathcal{P}}(G)$ are disjoint. A convex p-cover is called *convex p-partition* if it is a convex partition of a graph. Clearly, every graph G has convex 1-cover and convex n-cover. By Claude Berge [3], a set $S \subseteq X$ is a *clique* if every pair of vertices of S is adjacent in G. If $\boldsymbol{\mathcal{P}}_p(G)$ is a convex p-partition and all the sets of $\boldsymbol{\mathcal{P}}_p(G)$ are cliques, then $\boldsymbol{\mathcal{P}}_p(G)$ is called *clique p-partition* of graph G.

Definition 1. *A convex cover $\boldsymbol{\mathcal{P}}(G)$ of graph $G = (X; U)$ is called nontrivial convex cover if every set $Y \in \boldsymbol{\mathcal{P}}(G)$ satisfies the inequalities: $3 \leq |Y| \leq |X| - 1$. Consequently the elements of $\boldsymbol{\mathcal{P}}(G)$ are called nontrivial convex sets.*

Likewise, if a nontrivial convex cover $\boldsymbol{\mathcal{P}}(G)$ is a convex partition, we say that $\boldsymbol{\mathcal{P}}(G)$ is a *nontrivial convex partition.*

Definition 2. **[5]** *Convex cover number $\varphi_c(G)$ of a graph G is the least integer $p \geq 2$ for which G has a convex p-cover. Similarly, convex partition number $\theta_c(G)$ of a graph G is the least integer $p \geq 2$ for which G has a convex p-partition.*

Further, the least integer $p \geq 2$ for which graph G has a nontrivial convex p-cover is said to be *nontrivial convex cover number* $\varphi_{cn}(G)$. In the same way, the least integer $p \geq 2$ for which graph G has a nontrivial convex p-partition is said to be *nontrivial convex partition number* $\theta_{cn}(G)$.

Indeed, there are graphs for which there are no nontrivial convex p-covers or nontrivial convex p-partitions or both. For example, every convex simple graph has no nontrivial convex covers. A graph G is called *convex simple* if it does not contain nontrivial convex set [2].

Let us introduce the following notions.

Minimum convex cover $\boldsymbol{\mathcal{P}}_{\varphi_c}(G)$ is the convex p-cover of graph G such that $p = \varphi_c(G)$;

Minimum convex partition $\boldsymbol{\mathcal{P}}_{\theta_c}(G)$ is the convex p-partition of graph G such that $p = \theta_c(G)$;

Minimum nontrivial convex cover $\boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$ is the nontrivial convex p-cover of graph G such that $p = \varphi_{cn}(G)$;

Minimum nontrivial convex partition $\boldsymbol{\mathcal{P}}_{\theta_{cn}}(G)$ is the nontrivial convex p-partition of graph G such that $p = \theta_{cn}(G)$.

It is obvious that for any graph G we have $\varphi_c(G) \leq \theta_c(X)$. As above, if $\boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$ and $\boldsymbol{\mathcal{P}}_{\theta_{cn}}(G)$ exist, then $\varphi_{cn}(G) \leq \theta_{cn}(G)$. If $\boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$ exists, then $\varphi_c(G) \leq \varphi_{cn}(G)$. If $\boldsymbol{\mathcal{P}}_{\theta_{cn}}(G)$ exists, then $\theta_c(G) \leq \theta_{cn}(G)$.

Also, we introduce the following concept.

Definition 3. *A vertex $x \in X$ is called resident in $\boldsymbol{\mathcal{P}}(G)$ if x belongs to only one set of $\boldsymbol{\mathcal{P}}(G)$.*

By definition, every set of $\boldsymbol{\mathcal{P}}(G)$ contains at least one resident vertex in $\boldsymbol{\mathcal{P}}(G)$. If $\boldsymbol{\mathcal{P}}(G)$ is a convex partition of G, then all vertices of every set of $\boldsymbol{\mathcal{P}}(G)$ are resident in $\boldsymbol{\mathcal{P}}(G)$.

This paper is organized as follows. In Section 2 we describe some properties of minimum convex graph covers. In Section 3 we establish conditions for existence of graph G with given numbers $\varphi_c(G)$, $\theta_c(G)$ and $\varphi_{cn}(G)$, $\theta_{cn}(G)$. In Section 4 we prove that it is NP-complete to decide if a graph has a convex 2-cover. Deciding if a graph has convex 2-cover was declared an open problem in [5]. In addition, we prove that it is NP-complete to decide if a graph has nontrivial convex p-cover or nontrivial convex p-partition for $p \geq 2$.

2 Properties of minimum convex graph covers

Let $\boldsymbol{\mathcal{P}}_{\varphi_c}(G)$ be the minimum convex cover of a simple connected graph G.

Theorem 1. *If $\varphi_c(G) \geq 3$, then for every two sets $A, B \in \boldsymbol{\mathcal{P}}_{\varphi_c}(G)$, $A \neq B$, there exists $C \in \boldsymbol{\mathcal{P}}_{\varphi_c}(G) \backslash \{A, B\}$ such that there exist $a \in A$, $b \in B$, $c \in C \backslash (A \cup B)$, where $c \in \langle a, b \rangle$.*

Proof. Assume the converse. Suppose there exist sets $A, B \in \boldsymbol{\mathcal{P}}_{\varphi_c}(G)$, $A \neq B$, such that for all vertices $a \in A$, $b \in B$, we have $\langle a, b\rangle \subseteq A \cup B$. Thus, since $d - conv(A \cup B) = A \cup B$, we get the reduced convex cover number $\varphi_c(G)$. As $\varphi_c(G)$ is the least integer for which graph G has a convex p-cover, a contradiction follows. ■

Theorem 2. *If $\varphi_c(G) \geq 3$, then for each set $A \in \boldsymbol{\mathcal{P}}_{\varphi_c}(G)$, there exist $B, C \in \boldsymbol{\mathcal{P}}_{\varphi_c}(G)\backslash\{A\}$, $B \neq C$, such that there exist $a \in A\backslash(B \cup C)$, $b \in B$, $c \in C$, where $a \in \langle b, c\rangle$.*

Proof. Assume the converse. Suppose there exists a set $A \in \boldsymbol{\mathcal{P}}_{\varphi_c}(G)$ such that for every two sets $B, C \in \boldsymbol{\mathcal{P}}_{\varphi_c}(G)\backslash\{A\}$, $B \neq C$, we have $A \cap (\langle b, c\rangle\backslash(B \cup C)) = \emptyset$ for all vertices $b \in B$, $c \in C$. This yields that

$$d - conv(\bigcup_{S \in \boldsymbol{\mathcal{P}}_{\varphi_c}(G)\backslash\{A\}} S) = \bigcup_{S \in \boldsymbol{\mathcal{P}}_{\varphi_c}(G)\backslash\{A\}} S.$$

We obtain the convex 2-cover

$$\boldsymbol{\mathcal{P}}_2(G) = \boldsymbol{\mathcal{P}}_{\varphi_c}(G) = \{\bigcup_{S \in \boldsymbol{\mathcal{P}}_{\varphi_c}(G)\backslash\{A\}} S, A\}.$$

Finally, $\varphi_c(G) = 2$. This contradicts the condition of the theorem that $\varphi_c(G) \geq 3$. ■

Considering nontrivial convex cover as a particular case of convex cover, Theorems 1 and 2 have two consequences.

Corollary 1. *If $\varphi_{cn}(G) \geq 3$, then for every two sets $A, B \in \boldsymbol{\mathcal{P}}_{\varphi_{cn}(G)}$, $A \neq B$, there exists $C \in \boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)\backslash\{A, B\}$ such that there exist $a \in A$, $b \in B$, $c \in C\backslash(A \cup B)$, where $c \in \langle a, b\rangle$.*

Corollary 2. *If $\varphi_{cn}(G) \geq 3$, then for each set $A \in \boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$, there exist $B, C \in \boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)\backslash\{A\}$, $B \neq C$, such that there exist $a \in A\backslash(B \cup C)$, $b \in B$, $c \in C$, where $a \in \langle b, c\rangle$.*

Let $\alpha(G)$ be the vertex independence number of a graph G [3]. Next theorem is true.

Theorem 3. *Let $G = (X;\, U)$ be a simple connected graph and let S be a family of subsets of X with properties:*

a) $|S| \geq 2$;
b) each $Y \in S$ *is a clique;*
c) $X \setminus \bigcup_{Y \in S} Y$ *is not a clique;*
d) $Y \cap Z = \emptyset$ *for all* $Y, Z \in S$;
e) for each set $Y \in S$, *the equality* $\Gamma(y) = (Y \setminus \{y\}) \cup (X \setminus \bigcup_{Z \in S} Z)$ *is satisfied for every vertex* $y \in Y$.

Then the following conditions hold:

a) $\varphi_c(G) \geq \alpha(G)$, $\theta_c(G) \geq \alpha(G)$;
b) if $\boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$ *exists, then* $\varphi_{cn}(G) \geq \alpha(G)$;
c) if $\boldsymbol{\mathcal{P}}_{\theta_{cn}}(G)$ *exists, then* $\theta_{cn}(G) \geq \alpha(G)$;
d) every convex set of G *is a clique.*

Proof. Consider two nonadjacent vertices a, b of $X \setminus \bigcup_{Y \in S} Y$ and two vertices y, z such that $y \in Y$, $z \in Z$, where $Y, Z \in S$, $Y \neq Z$. Note that y, z are by definition nonadjacent.

From property e), it follows that $\bigcup_{Y \in S} Y \subseteq \langle a, b \rangle$ and $X \setminus \bigcup_{Y \in S} Y \subseteq \langle y, z \rangle$. Further, $X \setminus \bigcup_{Y \in S} Y \subseteq d - conv(\bigcup_{Y \in S} Y)$ and $\bigcup_{Y \in S} Y \subseteq d - conv(X \setminus \bigcup_{Y \in S} Y)$. Furthermore, we have $d - conv(\bigcup_{Y \in S} Y) = d - conv(X \setminus \bigcup_{Y \in S} Y) = X$. Thus, there is no convex set containing vertices a, b or y, z. This means that every convex set is a clique.

Let $M \subseteq X$ be the maximum independent set of G. In addition, from property e) it follows that $M \subseteq X \setminus \bigcup_{Y \in S} Y$, or $M \subseteq \bigcup_{Y \in S} Y$ such that every element of M belongs to exactly one set of S. By the above, every convex cover of graph G has at least $|M| = \alpha(G)$ sets. This implies the inequalities:

$$\varphi_c(G) \geq \alpha(G),\ \theta_c(G) \geq \alpha(G).$$

Moreover, if $\boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$ exists, then $\varphi_{cn}(G) \geq \alpha(G)$. Also, if $\boldsymbol{\mathcal{P}}_{\theta_{cn}}(G)$ exists, then $\theta_{cn}(G) \geq \alpha(G)$. ■

3 Existence of graphs with minimum convex covers

In this section several theorems regarding existence of simple connected graphs with fixed number of minimum convex covers and minimum convex partitions are proved.

For all $n \in N$, $n \geq 2$, there exists a graph on n vertices, which has a convex 2-cover or a convex 2-partition or both. For example, chain graph on $n \geq 2$ vertices has a convex 2-cover and a convex 2-partition. In addition, for every graph that has a nontrivial convex 2-cover the inequality $n \geq 4$ holds, because every set belonging to a convex 2-cover is nontrivial and has at least one resident vertex. On the other hand, for every graph that has a nontrivial convex 2-partition the inequality $n \geq 6$ is satisfied, because its sets are nontrivial and disjoint.

It is clear that for every graph G on n vertices, where $n = 2$ or $n = 3$, we have $\varphi_c(G) = \theta_c(G) = 2$.

First, we prove theorems regarding existence of graphs with fixed numbers $\varphi_c(G)$ and $\theta_c(G)$.

Theorem 4. *If G is a simple connected graph on $n \geq 4$ vertices, then $\varphi_c(G) \leq n - 2$.*

Proof. We distinguish two possible cases.

1) Let G be a graph that is not a convex simple graph. Suppose G has a nontrivial convex set S. Then, since $|S| \geq 3$, we obtain a convex p-cover of G such that $p = n - |S| + 1$. This convex p-cover consists of the set S and $n - |S|$ singletons (sets consisting of exactly one vertex). Substituting $|S| = n + 1 - p$ in $|S| \geq 3$, we get $p \leq n - 2$. So, $\varphi_c(G) \leq n - 2$.

2) Let G be a convex simple graph.

If $n = 4$, then G is a cycle. In this case, G has a convex 2-cover such that both convex sets consist of two adjacent vertices of G.

If $n \geq 5$, then graph G contains vertices x, y, such that $\Gamma(x) = \Gamma(y)$ and $|\Gamma(x)| \geq 3$ [2]. We choose two vertices u, v of $\Gamma(x)$. It is clear that u and v are nonadjacent, otherwise G is not a convex simple graph, because $\{x, u, v\}$ is a triangle, which is a nontrivial convex set. We get

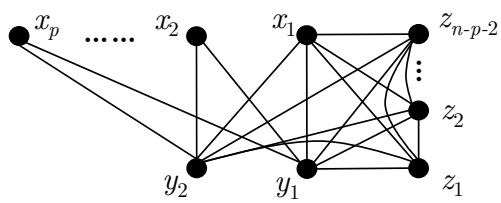

Figure 1. Graph G on n vertices such that $3 \leq p \leq n-2$, $\theta_c(G) = \varphi_c(G) = p$

a convex cover of G that consists of $p = n-2$ sets: $\{x, v\}$, $\{y, u\}$ and $p-4$ singletons.

By definition of $\varphi_c(G)$, we have $\varphi_c(G) \leq n-2$. ■

Corollary 3. *If G is a simple connected graph on $n \geq 4$ vertices, then $\theta_c(G) \leq n-2$.*

Theorem 5. *For any $p, n \in N$, $2 \leq p \leq n-2$, there exists a simple connected graph G on n vertices such that $\varphi_c(G) = p$.*

Proof. If $p = 2$, then take a chain graph G on n vertices for which $\varphi_c(G) = 2$.

If $p \geq 3$, we construct a graph $G = (X;\, U)$ as follows:

Step 1. let $X_1 = \{x_1, x_2, \ldots, x_p\}$, where any two vertices of X_1 are nonadjacent, i.e., X_1 is an independent set;

Step 2. if $p < n-2$, then define $X_2 = X_1 \cup Z$, where $Z = \{z_1, z_2, \ldots, z_{n-p-2}\}$ such that $Z \cup \{x_1\}$ is a clique. Otherwise, $X_2 = X_1$ and $Z = \emptyset$;

Step 3. $X = X_2 \cup \{y_1, y_2\}$, where $\Gamma(y_1) = \Gamma(y_2) = X_2$.

The resulted graph G is represented in Figure 1.

It is easy to verify that $|X| = n$.

Since X_1 is a maximum independent set in G, the independence number of this graph is $\alpha(G) = |X_1| = p$. The family $\{\{y_1\}, \{y_2\}\}$

satisfies the conditions of Theorem 3 in G. Thus, $\varphi_c(G) \geq p$ and every convex set of G is a clique.

It remains to show that there exists a convex p-cover of graph G.

Graph G has the convex p-cover $\boldsymbol{\mathcal{P}}_p(G)$ that consists of cliques $\{x_1, y_1\} \cup Z$, $\{x_2, y_2\}$, $\{x_3\}$, $\{x_4\}$,..., $\{x_p\}$.

So, since $\boldsymbol{\mathcal{P}}_p(G)$ is the convex p-cover of obtained graph G and $\varphi_c(G) \geq p$, it follows that $\boldsymbol{\mathcal{P}}_p(G)$ is the minimum convex cover of G and $\varphi_c(G) = p$. ■

Corollary 4. *For any $p, n \in N$, $2 \leq p \leq n-2$, there exists a simple connected graph G on n vertices such that $\theta_c(G) = p$.*

Further, a few theorems regarding existence of graphs with fixed numbers $\varphi_{cn}(G)$ and $\theta_{cn}(G)$ are proposed.

Theorem 6. *For any $p, n \in N$, $2 \leq p \leq \lfloor \frac{n}{3} \rfloor$, there exists a simple connected graph G on n vertices such that $\theta_{cn}(G) = p$.*

Proof. We construct a graph $G = (X; U)$ as follows:

Step 1. let $X_1 = \{x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}, \ldots, x_{p,1}, x_{p,2}\}$, where $x_{i,1} \sim x_{i,2}$ for $1 \leq i \leq p$;

Step 2. if $3p < n$, then define $X_2 = X_1 \cup Z$, where $Z = \{z_1, z_2, \ldots, z_{n-3p}\}$ such that $Z \cup \{x_{1,1}, x_{1,2}\}$ is a clique. Otherwise, $X_2 = X_1$ and $Z = \emptyset$;

Step 3. $X = X_2 \cup Y$, where $Y = \{y_1, y_2, \ldots, y_p\}$ such that $\Gamma(y_i) = X_2$ for $1 \leq i \leq p$.

The obtained graph G is represented in Figure 2.

It is easy to verify that $|X| = n$.

Since Y is a maximum independent set in G, the independence number of this graph is $\alpha(G) = |Y| = p$. The family $\{\{y_1\}, \{y_2\}, \ldots, \{y_p\}\}$ satisfies the conditions of Theorem 3 in G. Thus, $\theta_c(G) \geq p$ and if there exists $\boldsymbol{\mathcal{P}}_{\theta_{cn}}(G)$, then $\theta_{cn}(G) \geq p$. Also, every convex set of G is a clique.

It remains to show that there exists a nontrivial convex p-partition of graph G.

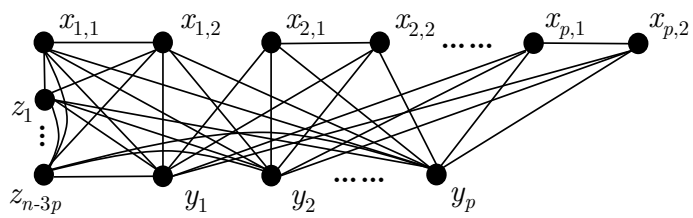

Figure 2. Graph G on n vertices such that $2 \leq p \leq \lfloor \frac{n}{3} \rfloor$, $\theta_{cn}(G) = p$

Graph G has a nontrivial convex p-partition $\boldsymbol{\mathcal{P}}_p(G)$ that consists of cliques $\{x_{1,1}, x_{1,2}, y_1\} \cup Z$, $\{x_{2,1}, x_{2,2}, y_2\}$, $\{x_{3,1}, x_{3,2}, y_3\}$,..., $\{x_{p,1}, x_{p,2}, y_p\}$.

So, since $\boldsymbol{\mathcal{P}}_p(G)$ is a nontrivial convex p-partition of obtained graph G and $\theta_{cn}(G) \geq p$, it follows that $\boldsymbol{\mathcal{P}}_p(G)$ is the minimum nontrivial convex partition of G and $\theta_{cn}(G) = p$. ■

Let C_4 be a cycle on 4 vertices.

Theorem 7. *If G is a simple connected graph on 4 vertices, then $\varphi_{cn}(G) = 2$ if and only if $G \neq C_4$.*

Proof. By definition of nontrivial convex cover, G has a nontrivial convex p-cover if and only if $p = 2$. In Figure 3 simple connected graphs on 4 vertices are represented. It can be easily checked that every graph from Figure 3, except the cycle C_4, has a nontrivial convex 2-cover. Now, if we recall that nontrivial convex cover number is the least integer $p \geq 2$ for which graph G has a nontrivial convex p-cover, we get $\varphi_{cn}(G) = 2$ for every simple connected graph G on 4 vertices, where $G \neq C_4$. ■

Theorem 8. *If G is a simple connected graph on $n \geq 5$ vertices, then $\varphi_{cn}(G) < n - 2$.*

Proof. There is no a nontrivial convex n-cover or a nontrivial convex $(n-1)$-cover, because every convex set of nontrivial convex cover has

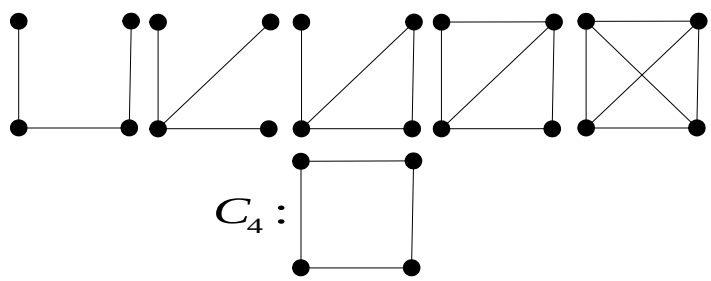

Figure 3. All simple connected graphs on 4 vertices

at least one resident vertex in this convex cover and contains at least three vertices.

Let us prove that for every graph G the inequality $\varphi_{cn}(G) < n-2$ is satisfied. The proof is by reductio ad absurdum. We can assume without loss of generality that there exists a graph $G = (X\ ,U)$ such that $\varphi_{cn}(G) = n-2$. It is required that $n \geq 5$, consequently $\varphi_{cn}(G) \geq 3$. Let $\boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$ be the minimum nontrivial convex cover of G. In this case, every set $S \in \boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$ satisfies equality $|S| = 3$ and contains exactly one resident vertex in $\boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$. Further, there are two vertices $x, y \in X$, which are common for all sets of $\boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$. Notice that $x \sim y$, otherwise connectivity of nontrivial convex sets of $\boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$ implies $\Gamma(x) = \Gamma(y) = X\backslash\{x, y\}$ and furthermore $d - conv(\{x, y\}) = X$. According to Corollaries 1 and 2, all vertices of set $X\backslash\{x, y\}$ are nonadjacent in G. Finally, we obtain a nontrivial convex 2-cover $\boldsymbol{\mathcal{P}}_2(G) = \boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G) = \{\{x, y, z\}, X\backslash\{z\}\}$, where $z \in X\backslash\{x, y\}$. Thus, $\varphi_{cn}(G) = 2$. This contradiction concludes the proof. ■

Theorem 9. *For any $p, n \in N$, $2 \leq p \leq n-3$, there exists a simple connected graph G on n vertices such that $\varphi_{cn}(G) = p$.*

Proof. We construct a graph $G = (X;\ U)$ as follows:

Step 1. let $X_1 = \{x_1, x_2, \ldots, x_p\}$, where all vertices of X_1 are nonadjacent, i.e., X_1 is an independent set;

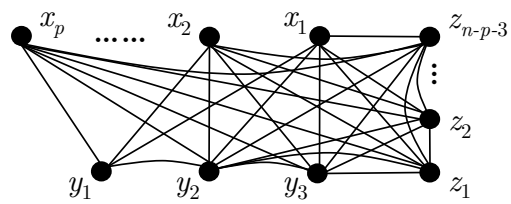

Figure 4. Graph G on n vertices such that $2 \leq p \leq n-3$, $\varphi_{cn}(G) = p$

Step 2. if $p < n-3$, then define $X_2 = X_1 \cup Z$, where $Z = \{z_1, z_2, \ldots, z_{n-p-3}\}$ such that $Z \cup \{x\}$ is a clique for all $x \in X_1$. Otherwise, $X_2 = X_1$ and $Z = \emptyset$;

Step 3. $X = X_2 \cup Y$, where $Y = \{y_1, y_2, y_3\}$ such that $\Gamma(y_1) = X_1 \cup \{y_2\}$, $\Gamma(y_2) = X_2 \cup \{y_1, y_3\}$ and $\Gamma(y_3) = X_2 \cup \{y_2\}$.

The obtained graph G is represented in Figure 4.

It is easy to verify that $|X| = n$.

Since X_1 is a maximum independent set in G, the independence number of this graph is $\alpha(G) = |X_1| = p$. The family $\{\{x_1\}, \{x_2\}, \ldots, \{x_p\}\}$ satisfies the conditions of Theorem 3 in G. Thus, $\varphi_c(G) \geq p$ and if there exists $\boldsymbol{\mathcal{P}}_{\varphi_{cn}}(G)$, then $\varphi_{cn}(G) \geq p$. Also, every convex set of G is a clique.

It remains to show that there exists a nontrivial convex p-cover of graph G.

Graph G has a nontrivial convex p-cover $\boldsymbol{\mathcal{P}}_p(G)$ that consist of cliques $\{x_1, y_2, y_3\} \cup Z$, $\{x_2, y_1, y_2\}$, $\{x_3, y_1, y_2\}$, $\ldots$, $\{x_p, y_1, y_2\}$.

So, since $\boldsymbol{\mathcal{P}}_p(G)$ is a nontrivial convex p-cover of obtained graph G and $\varphi_{cn}(G) \geq p$, it follows that $\boldsymbol{\mathcal{P}}_p(G)$ is the minimum nontrivial convex cover of G and $\varphi_{cn}(G) = p$. ■

4 NP-completeness

Let us examine the complexity of convex cover problems.

Deciding whether a graph G has a convex p-cover, for $p \geq 2$, is said to be convex p-cover problem. Similarly, deciding whether a graph G has a convex p-partition, for $p \geq 2$, is said to be convex p-partition problem. In the same way, we introduce nontrivial convex p-cover and nontrivial convex p-partition problems, where nontrivial convex covers and nontrivial convex partitions are considered.

It was shown in [4], [6] that the convex p-partition problem is NP-complete for $p \geq 2$. Also, we know that the convex p-cover problem is NP-complete for $p \geq 3$ [5]. Deciding if a graph has a convex 2-cover was declared an open problem in the paper [5].

We prove that the convex 2-cover problem is NP-complete.

The complexity of this case is proved by reducing the NP-complete 1-IN-3 3 SAT problem [8] to a convex 2-cover problem.

1-*IN*-3 3 *SAT problem:*

Instance: Set $V = \{v_1, v_2, \ldots, v_n\}$ of variables, collection $\mathcal{C} = \{\mathcal{c}_1, \mathcal{c}_2, \ldots, \mathcal{c}_m\}$ of clauses over V such that each clause $\mathcal{c} \in \mathcal{C}$ has $|\mathcal{c}| = 3$ and no negative literals.

Question: Is there a truth assignment for V such that each clause in $\mathcal{C}$ has exactly one true literal?

We say that $\mathcal{C}$ is *satisfiable* if there exists a truth assignment for V such that $\mathcal{C}$ is satisfiable and each clause in $\mathcal{C}$ has exactly one true variable.

Theorem 10. *The convex* 2*-cover problem is NP-complete.*

Proof. We mention that this problem is in NP, because verifying if a set is convex can be done in polynomial time [7]. Further, we reduce 1-IN-3 3 SAT to the convex 2-cover problem. First, we determine the structure of a particular graph $G = (X; U)$ for a convex 2-cover from a generic instance $(V, \mathcal{C})$ of 1-IN-3 3 SAT. Next, we prove that $\mathcal{C}$ is satisfiable if and only if G has a convex 2-cover. For this purpose, we prove that a convex 2-cover of G defines a truth assignment that satisfies $(V, \mathcal{C})$. At the same time, we prove that a truth assignment that satisfies $(V, \mathcal{C})$ defines a convex 2-cover of G.

Let graph G be given by vertex set X and edge set U.

The vertex set X consists of:

a) $\boldsymbol{\mathcal{V}} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n\}$, $Y = \{f, y_1, y_2, y_3, y_4, y_5\}$, $Z = \{t, z_1, z_2, z_3, z_4, z_5\}$;
b) $F = \{f_j | 1 \le j \le m\}$, $T = \{t_j | 1 \le j \le m\}$;
c) $L = \{l_j^i | 1 \le j \le m, 1 \le i \le 3\}$, $\boldsymbol{\mathcal{L}} = \{\boldsymbol{\ell}_j^i | 1 \le j \le m, 1 \le i \le 3\}$, $Q = \{q_j^i | 1 \le j \le m, 1 \le i \le 3\}$;

We get $X = \boldsymbol{\mathcal{V}} \cup Y \cup Z \cup F \cup T \cup L \cup Q \cup \boldsymbol{\mathcal{L}}$. Every variable $v_i \in V$ corresponds to vertex $\boldsymbol{u}_i \in \boldsymbol{\mathcal{V}}$. Every clause $\boldsymbol{c}_j \in \boldsymbol{\mathcal{C}}$ corresponds to eleven vertices: f_j, l_j^1, l_j^2, l_j^3, $\boldsymbol{\ell}_j^1$, $\boldsymbol{\ell}_j^2$, $\boldsymbol{\ell}_j^3$, q_j^1, q_j^2, q_j^3, t_j.

The edge set U satisfies conditions:

a) $\boldsymbol{\mathcal{V}} \cup Q$ is a clique in G;
b) $\Gamma(f) = \boldsymbol{\mathcal{V}} \cup Q \cup F \cup \{y_3, y_4\}$ and $\Gamma(t) = \boldsymbol{\mathcal{V}} \cup Q \cup T \cup \{z_3, z_4\}$;
c) $\Gamma(y_5) = F \cup \{y_3, y_4\}$ and $\Gamma(z_5) = T \cup \{z_3, z_4\}$;
d) there exist the following edges: $\{y_1, y_3\}$, $\{y_1, y_4\}$, $\{y_2, y_3\}$, $\{y_2, y_4\}$, $\{z_1, z_3\}$, $\{z_1, z_4\}$, $\{z_2, z_3\}$, $\{z_2, z_4\}$;
e) every clause $\boldsymbol{c}_j = \{v_a, v_b, v_c\}$, $1 \le j \le m$, corresponds to eighteen edges: $\{l_j^1, v_a\}$, $\{l_j^2, v_b\}$, $\{l_j^3, v_c\}$, $\{l_j^1, f_j\}$, $\{l_j^2, f_j\}$, $\{l_j^3, f_j\}$, $\{\boldsymbol{\ell}_j^1, t_j\}$, $\{\boldsymbol{\ell}_j^2, t_j\}$, $\{\boldsymbol{\ell}_j^3, t_j\}$, $\{q_j^1, \boldsymbol{\ell}_j^1\}$, $\{q_j^2, \boldsymbol{\ell}_j^2\}$, $\{q_j^3, \boldsymbol{\ell}_j^3\}$, $\{l_j^1, \boldsymbol{\ell}_j^2\}$, $\{l_j^1, \boldsymbol{\ell}_j^3\}$, $\{l_j^2, \boldsymbol{\ell}_j^1\}$, $\{l_j^2, \boldsymbol{\ell}_j^3\}$, $\{l_j^3, \boldsymbol{\ell}_j^1\}$, $\{l_j^3, \boldsymbol{\ell}_j^2\}$.

We skip the trivial case $|\boldsymbol{\mathcal{C}}| = 1$ of 1-IN-3 3 SAT problem. Let us consider $|\boldsymbol{\mathcal{C}}| \ge 2$.

If $G = (X; U)$ has a convex 2-cover, then $\boldsymbol{\mathcal{C}}$ is satisfiable.

Let $\boldsymbol{\mathcal{P}}_2(G) = \{S_f, S_t\}$ be a convex 2-cover of G. For every $i, j \in \{1, 2\}$ we have $d - conv(\{y_i, z_j\}) = X$.

Let $y_1, y_2 \in S_f$, $z_1, z_2 \in S_t$ and let $S_1 = \{y_3, y_4, y_5, f\} \cup F$, $S_2 = \{z_3, z_4, z_5, t\} \cup T$.

Let us enumerate some properties:

Property 1: $S_1 \cap S_t = \emptyset$ and $S_2 \cap S_f = \emptyset$.

We notice what $S_1 \subseteq d - conv(\{y_1, y_2\})$, $S_2 \subseteq d - conv(\{z_1, z_2\})$. Consequently we have $S_1 \subseteq S_f$, $S_2 \subseteq S_t$.

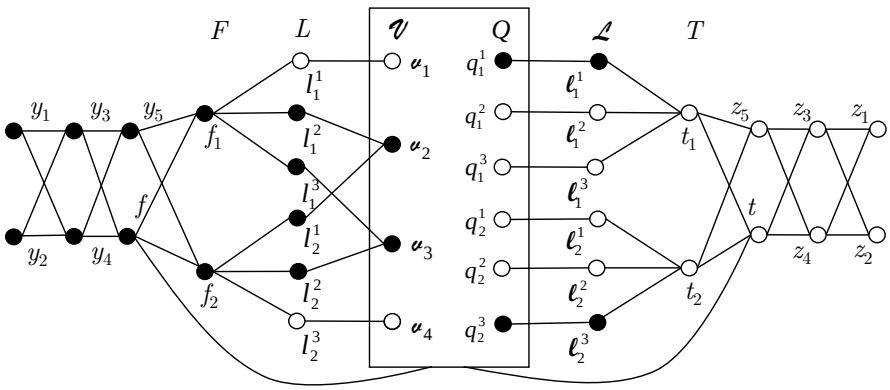

Figure 5. The convex 2-cover of the graph G for the instance $(V, \mathcal{C}) = (\{v_1, v_2, v_3, v_4\}, \{\{v_1, v_2, v_3\}, \{v_2, v_3, v_4\}\})$

Moreover, for each $u \in F \cup \{f\}$, we get $d - conv(\{u, t\} \cup T) \subseteq d - conv(\{f\} \cup F \cup S_2) \subseteq d - conv(S_1 \cup S_2) \subseteq d - conv(S_f \cup S_t) = X$. This implies that $u \notin S_t$ for each $u \in F \cup \{f\}$. Similarly, for each $u \in T \cup \{t\}$, we get $d - conv(\{u, f\} \cup F) \subseteq d - conv(\{t\} \cup T \cup S_1) \subseteq d - conv(S_1 \cup S_2) \subseteq d - conv(S_f \cup S_t) = X$. This implies that $u \notin S_f$ for each $u \in T \cup \{t\}$. Thus, $S_1 \cap S_t = \emptyset$ and $S_2 \cap S_f = \emptyset$.

Property 2: Sets $L, \mathcal{V}, Q, \mathcal{L}$ are uniquely interdependent.

If vertex l_j^i belongs to S_t, then $\Gamma(l_j^i) \cap \mathcal{V} \subseteq S_t$ and $\boldsymbol{\ell}_j^k$ belongs to S_t for $1 \leq k \leq 3$, $k \neq i$.

If vertex $\boldsymbol{u}_i$ belongs to S_t, then $\Gamma(\boldsymbol{u}_i) \cap L \subseteq S_t$ and for all $l_j^a \in \Gamma(\boldsymbol{u}_i) \cap L$ vertices $\boldsymbol{\ell}_j^k$ belong to S_t for $1 \leq k \leq 3$, $k \neq a$.

Vertex $\boldsymbol{\ell}_j^i$ belongs to S_f if and only if q_j^i belongs to S_f. If vertex $\boldsymbol{\ell}_j^i$ belongs to S_f, then $L' = \{l_j^k | 1 \leq k \leq 3, k \neq i\} \subseteq S_f$ and $\Gamma(l_j^k) \cap \mathcal{V}$ is contained in S_f for all $l_j^k \in L'$.

Property 3: Exactly one vertex of $L_j = \{l_j^1, l_j^2, l_j^3\}$ belongs to S_t, for $1 \leq j \leq m$, and exactly one vertex of $\mathcal{L}_j = \{\boldsymbol{\ell}_j^1, \boldsymbol{\ell}_j^2, \boldsymbol{\ell}_j^3\}$ belongs to S_f, for $1 \leq j \leq m$.

Exactly one vertex of every set $L_j = \{l_j^1, l_j^2, l_j^3\}$, $1 \le j \le m$, belongs to S_t. In the converse case, if two vertices $\{l_j^a, l_j^b\}$ of L_j belong to S_t, then f_j belongs to S_t. By Property 1, we get a contradiction. If no vertex of $L_j = \{l_j^1, l_j^2, l_j^3\}$ belongs to S_t, then $L_j \subseteq S_f$, $\boldsymbol{\mathcal{L}}_j = \{\boldsymbol{\ell}_j^1, \boldsymbol{\ell}_j^2, \boldsymbol{\ell}_j^3\} \subseteq S_f$ and t_j belongs to S_f. Now by Property 1, we have a contradiction.

In the same way, exactly one vertex of every set $\boldsymbol{\mathcal{L}}_j = \{\boldsymbol{\ell}_j^1, \boldsymbol{\ell}_j^2, \boldsymbol{\ell}_j^3\}$, $1 \le j \le m$, belongs to S_f.

We associate $\boldsymbol{\mathcal{V}}$ with V and L with $\boldsymbol{\mathcal{C}}$ such that convex 2-cover represents a truth assignment for $\boldsymbol{\mathcal{V}}$, where the variable v_i is true if and only if the vertex $\boldsymbol{\alpha}_i \in S_t$.

It follows from Properties 1, 2 and 3 that if G has a convex 2-cover $\boldsymbol{\mathcal{P}}_2(G) = \{S_f, S_t\}$, then $\boldsymbol{\mathcal{C}}$ is satisfiable. Let us remark that sets S_f, S_t are nontrivial and disjoint.

If $\boldsymbol{\mathcal{C}}$ is satisfiable, then $G = (X;\, U)$ has a 2-convex cover.

Suppose that there exists a truth assignment which satisfies $(V, \boldsymbol{\mathcal{C}})$. We construct a convex 2-cover $\boldsymbol{\mathcal{P}}_2(G) = \{S_f, S_t\}$ as follows:

Step 1. Define $S_t = \{z_1, z_2, z_3, z_4, z_5, t\} \cup T$;
Step 2. For each true variable v_i of V we add vertex $\boldsymbol{\alpha}_i$ and the set $L' = \Gamma(\boldsymbol{\alpha}_i) \cap L$ to S_t and for each $l_j^a \in L'$ we add vertices q_j^b, $\boldsymbol{\ell}_j^b$ to S_t such that $\boldsymbol{\ell}_j^b \sim l_j^a$ and $q_j^b \sim \boldsymbol{\ell}_j^b$;
Step 3. Define $S_f = X \backslash S_t$.

Clearly, for the resulting convex 2-cover $\boldsymbol{\mathcal{P}}_2(G) = \{S_f, S_t\}$ Properties 1, 2 and 3 are satisfied. Hence, if $\boldsymbol{\mathcal{C}}$ is satisfiable, then G has convex 2-cover. Note also that the sets S_f and S_t are nontrivial and disjoint.

In Figure 5 the graph G, which corresponds to a particular instance $(V, \boldsymbol{\mathcal{C}}) = (\{v_1, v_2, v_3, v_4\}, \{\{v_1, v_2, v_3\}, \{v_2, v_3, v_4\}\})$ is represented. Sets $Q \cup \boldsymbol{\mathcal{V}} \cup \{f\}$ and $Q \cup \boldsymbol{\mathcal{V}} \cup \{t\}$ generate cliques in G. White vertices belong to S_t and black vertices belong to S_f. The white vertices of $\boldsymbol{\mathcal{V}}$ represent the variables of V set to true. All edges between L and $\boldsymbol{\mathcal{L}}$ are represented in Figure 6. ■

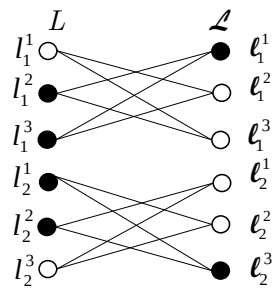

Figure 6. Edges between L and $\mathcal{L}$

Now if we recall that convex p-cover problem is NP-complete, for $p \geq 3$, we can affirm that convex p-cover problem is NP-complete for $p \geq 2$.

Corollary 5. *Convex 2-partition problem, nontrivial convex 2-cover problem and nontrivial convex 2-partition problem are NP-complete.*

Proof. By construction in the previous theorem of a particular graph $G = (X;\, U)$ for convex 2-cover problem from a generic instance $(V, \mathcal{C})$ of 1-IN-3 3 SAT problem, we conclude that G can be covered only by nontrivial disjoint convex sets. Hence, every convex 2-cover is a convex 2-partition in G. Moreover, every convex 2-cover is a nontrivial convex 2-cover in G. ■

Taking into account Theorem 10 and Corollary 5, we affirm that Theorem 10 is stronger than Theorem 4 in [6], which proves only NP-completeness of convex 2-partition problem.

Furthermore, we prove that nontrivial convex p-cover problem, for $p \geq 3$, is NP-complete. We reduce NP-complete clique p-partition problem, for $p \geq 3$ [9], to a nontrivial convex p-cover problem.

Clique p-partition problem:
Instance: Graph $G = (X;\, U)$ and $p \in N$, $p \geq 3$.
Question: Is there a partition of X into p disjoint cliques?

Theorem 11. *The nontrivial convex p-partition problem is NP-complete for $p \geq 3$.*

Proof. The problem is in NP, because determining if a set is convex can be done in polynomial time [7].

Let $G = (X; U)$ be a generic graph of clique p-partition problem. Without loss of generality, it can be assumed that X is not a clique. We obtain a particular graph $G' = (X'; U')$ of nontrivial convex p-partition problem from G by adding auxiliary sets $Y = \{y_1, y_2, \ldots, y_p\}$ and $Z = \{z_1, z_2, \ldots, z_p\}$ to X such that $X' = X \cup Y \cup Z$, where $\Gamma(y_i) = X \cup \{z_i\}$ and $\Gamma(z_i) = X \cup \{y_i\}$ for $1 \leq i \leq p$.

Graph G' satisfies the conditions of Theorem 3. Thus, every convex set of G' is a clique.

If $\boldsymbol{\mathcal{P}}_p(G)$ is a clique p-partition of G, $p \geq 3$, then we obtain a nontrivial convex p-partition $\boldsymbol{\mathcal{P}}_p(G')$ of G' by addition of set $\{y_i, z_i\}$ to X_i, where $X_i \in \boldsymbol{\mathcal{P}}_p(G)$, for $1 \leq i \leq p$.

On the other hand, a nontrivial convex p-partition $\boldsymbol{\mathcal{P}}_p(G')$ of G', $p \geq 3$, implies existence of a clique p-partition $\boldsymbol{\mathcal{P}}_p(G)$ of G by subtraction of set $\{y_i, z_i\}$ from X'_i , where $X'_i \in \boldsymbol{\mathcal{P}}_p(G')$, for $1 \leq i \leq p$. ■

Corollary 6. *The nontrivial convex p-cover problem is NP-complete for $p \geq 3$.*

Proof. The problem is also in NP, because determining if a set is convex can be done in polynomial time [7].

We know that any proper convex set of graph G' constructed in previous theorem, is a clique. Let $\boldsymbol{\mathcal{P}}_p(G')$ be a nontrivial convex p-cover of G'. We get a family of sets $\boldsymbol{\mathcal{P}} = \{X_1, X_2, \ldots, X_p\}$ such that $X_i = X'_i \backslash \{y_i, z_i\}$, where $X'_i \in \boldsymbol{\mathcal{P}}_p(G')$ for $1 \leq i \leq p$. Removing from $\boldsymbol{\mathcal{P}}$ all sets contained in the union of other sets of the family $\boldsymbol{\mathcal{P}}$ we obtain a convex k-partition $\boldsymbol{\mathcal{P}}_k(G)$ of G such that $k \leq p$, where G is a graph of clique p-partition problem. Note also that if any graph has a clique q-partition and there exists a set S of this partition that is not a singleton, then dividing S into two cliques, we get a clique $(q+1)$-partition. Thus, G has a clique p-partition.

On the other hand, we know from previous theorem that every clique p-partition of G, $p \geq 3$, implies existence of nontrivial convex

p-partition of G'. Now, if we recall that every nontrivial convex p-partition is a nontrivial convex p-cover, we deduce that every clique p-partition of G, $p \geq 3$, implies existence of nontrivial convex p-cover of G'. ■

We affirm that nontrivial convex p-cover problem and nontrivial convex p-partition problem are NP-complete for $p \geq 2$. Indeed, this follows from Theorems 11 and from Corollaries 5 and 6.

5 Conclusion

We prove that the problem of deciding if a graph has a convex 2-cover is NP-complete. Since Theorem 10 proves NP-completeness of convex 2-cover problem and Corollary 5, as consequence of Theorem 10, proves NP-completeness of convex 2-partition problem, we conclude that Theorem 10 is stronger than Theorem 4 in [6], which proves only NP-completeness of convex 2-partition problem. We affirm that convex p-cover problem and convex p-partition problem are NP-complete for $p \geq 2$.

Also, we prove that it is NP-complete to decide if a graph has a nontrivial convex p-cover or nontrivial convex p-partition for $p \geq 2$.

We discover some properties of minimum convex covers and minimum convex partitions of graphs. We establish conditions for existence of graph G with given numbers $\varphi_c(G)$, $\theta_c(G)$ and $\varphi_{cn}(G)$, $\theta_{cn}(G)$.

References

[1] V. Bolteansky, P.Soltan. *Combinatorial geometry of the various classes of convex sets.* Chişinău, 1978. (in Russian)

[2] S. Cataranciuc, N.Sur. *D-convex simple and quasi-simple graphs.* Chişinău, Republic of Moldova, 2009. (in Romanian)

[3] C. Berge. *Graphs and Hypergraphs.* New York: Elsevier, 1973.

[4] D. Artigas, M.C.Dourado, J.L.Szwarcfiter. *Convex Partitions of Graphs.* Electronic Notes in Discrete Mathematics, vol. 29 (2007), pp. 147–151.

[5] D. Artigas, S.Dantas, M.C.Dourado, J.L.Szwarcfiter. *Convex covers of graphs.* Matemática Contemporânea, Sociedade Brasileira de Matemática, vol. 39 (2010), pp. 31–38.

[6] D. Artigas, S.Dantas, M.C.Dourado, J.L.Szwarcfiter. *Partitioning a graph into convex sets.* Discrete Mathematics, vol. 311 (2011), pp. 1968–1977.

[7] M.C. Dourado, J.G.Gimbel, F.Protti, J.L.Szwarcfiter. *On the computation of the hull number of a graph.* Discrete Mathematics, vol. 309 (2009), pp. 5668–5674.

[8] T. J. Schaefer. *The complexity of satisfiability problems.* Proceeding STOC '78 Proceedings of the tenth annual ACM symposium on Theory of computing, ACM New York, NY, USA , 1978, pp. 216–226.

[9] R. Karp. *Reducibility among combinatorial problems.* In: R.E. Miller and J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum, New York, 1972, pp. 85–103.

Radu Buzatu, Sergiu Cataranciuc Received July 27, 2015

Radu Buzatu
State University of Moldova
60 A. Mateevici, MD-2009, Chişinău, Republic of Moldova
E–mail: radubuzatu@gmail.com

Sergiu Cataranciuc
State University of Moldova
60 A. Mateevici, MD-2009, Chişinău, Republic of Moldova
E–mail: s.cataranciuc@gmail.com

Computer Science Journal of Moldova, vol.**23**, no.3(69), 2015

Set-theoretic Analysis of Nominative Data*

Volodymyr G. Skobelev, Ievgen Ivanov, Mykola Nikitchenko

Abstract

In the paper we investigate the notion of nominative data that can be considered as a general mathematical model of data used in computing systems. The main attention is paid to flat nominative data called nominative sets. 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 proposed. We investigate a lower semilattice of nominative sets in terms of lower and upper cones, closed and maximal closed intervals of nominative sets. The obtained results can be used in formal software development.

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

1 Introduction

The significance of the problem of elaborating the theory of programming and linking it with software development practice was recognized by many researchers [1–6], and in particular, it was mentioned as one of the grand challenges in computing by T. Hoare in his influential talk "The Verifying Compiler: a Grand Challenge for computing research of the 21st century" [7]: *"To build the link between the theory of programming and the products of software engineering practice is still a grand challenge for scientific research in computing; the development*

* This work was partially supported by the research project No. 11BF015-02 "Formal specifications and methods of development of reliable software systems", Taras Shevchenko National University of Kyiv, Ukraine.

of a verifying compiler is an essential tool and target for this research; it will make the results of the research available to software engineers of the future, and so contribute to the quality and reliability of all the programs that they produce."

Currently there exist various approaches that try to deal with this global problem [2–6,8–11], each of which has its own methodology. This paper advocates the so-called *composition-nominative approach* to program formalization [12–14]. The starting point of this approach is the view of software as a data processor that must deal with many forms of data used in computing systems (e.g. arrays, lists, dictionaries, tables, trees, etc.). Thus for solving the mentioned grand problem, firstly one needs to develop a unified, adequate, and tractable theoretical model of data that can serve as a basis for building adequate semantic models of programming language constructs and programs.

The unified data model proposed in the composition-nominative approach is called *nominative data* [12, 15, 16] and is based on the name-value relation. In the simplest case one can view a nominative data as a collection of associations between names and values that can be denoted as $[name_1 \mapsto value_1, name_2 \mapsto value_2, ...]$, or, more formally, as a partial function from the set of all possible names to the set of all possible values.

The following simple example illustrates this notion. In most web applications (e.g. online reservation systems, online stores, search engines, etc.) the primary method of obtaining information from a user is based on web forms. A typical web form consists of several named mandatory and optional fields, e.g. Fig. 1.

A natural mathematical model of a user-supplied data in this case is a partial function that maps field names to the corresponding filled values, assuming that this function is undefined on all unfilled fields. This partial function is a nominative data of a particular type called a *nominative set*. For example, for a web form with mandatory fields $FirstName$, $LastName$, $Email$, $Country$, $Organization$ and an optional field $WebSite$, a data instance provided by the user can be modeled as a nominative set (partial function)

$$d : \{FirstName, LastName, Email, Country,$$

$$Organization, WebSite\} \to A,$$

where A denotes the set of all possible field values (strings).

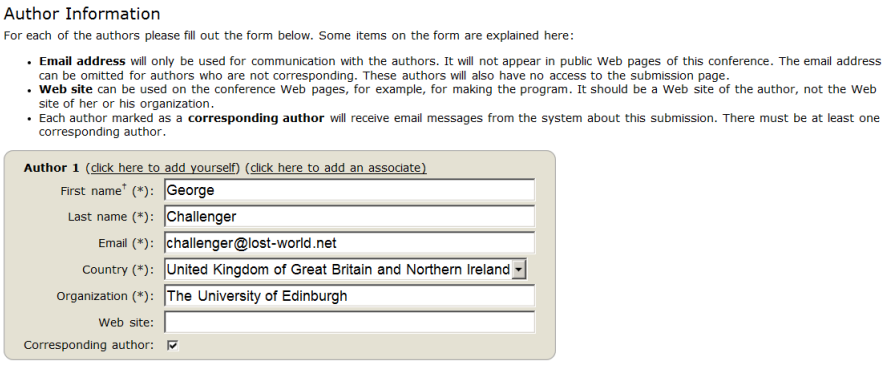

Figure 1. A screenshot of the EasyChair conference management system [17] (`http://www.easychair.org`). An example of a web form with mandatory and optional fields.

If $d(WebSite)$ is undefined, this means that the user left the corresponding field unfilled. The values of such a function can be conveniently specified using a notation of the form

$$[FieldName1 \mapsto FieldValue1, FieldName2 \mapsto FieldValue2, ...],$$

e.g.,

$$[FirstName \mapsto George, LastName \mapsto Challenger,$$
$$Email \mapsto challenger@lost\text{-}world.net, Country \mapsto the\ UK,$$
$$Organization \mapsto The\ University\ of\ Edinburgh].$$

Clearly, development of the composition-nominative approach requires refinement of the idea of data as name-value relations. Very basic questions concerning the nature of name-value relations already give hints concerning the possible directions of this refinement:

- Are the names unstructured (simple) or structured (complex), e.g. strings in a certain alphabet ?

- Are the values unstructured (simple) or structured (complex), e.g. can values be nominative data themselves ?
- Is only direct naming possible (i.e. values cannot be names) ? Or indirect naming is also allowed (values can be names) ?

Different answers to these questions lead to different types of nominative data (TND1–TND8) [15] illustrated in Fig. 2.

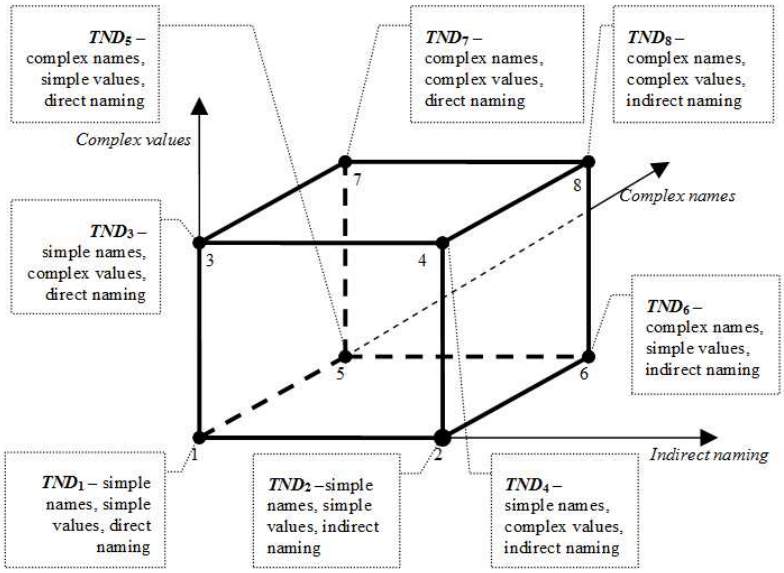

Figure 2. Types of nominative data

Although the idea behind nominative data is intuitively clear, the absence of unique answers to even such basic questions makes the process of their formalization and application to program semantics and the problems of software specification and verification non-trivial. In fact, one has to consider and investigate many formalizations of nominative data and study their suitability for different classes of problems. Another complication is that algebraic structures that arise from sets of

nominative data of different types turn out to be different from structures traditionally considered in algebra and computer science (e.g. rings, lattices, boolean algebras, etc.) and currently remain virtually unstudied.

To deal effectively with complexity of data and program formalization, composition-nominative approach proposes the following principles [12]:

- *Development principle* (from abstract to concrete): the process of development of program notions must start from abstract understanding and proceed to more concrete considerations.
- *Principle of integrity of intensional and extensional aspects*: program notions should be presented in the integrity of their intensional and extensional aspects, but the intensional aspects play a leading role.
- *Principle of priority of semantics over syntax*: semantic and syntactical aspects of programs should be first studied separately, and then in their integrity in which semantic aspects prevail over syntactical aspects.
- *Compositionality principle*: programs are constructed from simpler programs using operations called compositions which represent semantics of programming language constructs.
- *Nominativity principle*: naming relations are the basic ones in constructing data and programs.

These principles are applied for constructing a hierarchy of program models of various levels of abstraction and generality with the general aim of providing a mathematical basis for development of formal methods of analysis and synthesis of reliable software systems.

Above mentioned principles are applied to program formalization as follows:

- Data in computing systems are formalized as specific classes of nominative data. A set of nominative data of a particular type

together with the basic operations on these data forms an algebra called a data algebra.

- Programs that operate on data are formalized as partial functions that map nominative data to nominative data, also called (bi-) nominative functions.

- Program combination operators (e.g. sequential execution, branching, cycle, etc.) are formalized as operations (also called compositions) that map (bi-)nominative functions to (bi-)nominative function. A set of programs (modeled as nominative functions) that can be obtained from basic operations on data using compositions together with compositions forms an algebra (program algebra) that represents compositional semantics of a programming language. Proving program properties is done by proving certain facts in a program algebra.

In this paper we will study the basic type of nominative data TND1, or data with unstructured names and unstructured values, also called nominative sets. In particular, we will investigate rich algebraic structures that arise from it.

This paper is organized as follows: in Section 2 we give rigorous definitions of nominative sets and other associated notions; in Section 3 we define the basic operations on nominative sets by analogy with set-theoretic operations and study algebraic systems that arise from these definitions; in Section 4 we investigate a partial ordering on nominative sets and the associated poset using lattice theory [18–20]; in Section 5 we give conclusions.

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 $\mathfrak{F}_{V,A}$:

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

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 "$\backslash$" to the set $\mathfrak{F}_{V,A}$. Indeed, for any $f_1, f_2, f \in \mathfrak{F}_{V,A}$ we can define:

$$f_1 \cap f_2 = f \Leftrightarrow graph(f_1) \cap graph(f_2) = graph(f), \tag{2}$$
$$f_1 \backslash f_2 = f \Leftrightarrow graph(f_1) \backslash graph(f_2) = graph(f).$$

Obviously, for any $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following formulas hold:

$$\mathrm{Dom}(f_1 \cap f_2) \subseteq \mathrm{Dom} f_1 \cap \mathrm{Dom} f_2,$$

$$f_1|_X \cap f_2|_Y = (f_1 \cap f_2)|_{X\cap Y} \ (X, Y \subseteq V),$$
$$\mathrm{Dom} f_1 \backslash \mathrm{Dom} f_2 \subseteq \mathrm{Dom}(f_1 \backslash f_2) \subseteq \mathrm{Dom} f_1.$$

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* $(\mathfrak{F}_{V,A}, \backslash)$ *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:

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

and

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

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

$$[f_1, f_2] \in \mathfrak{S}^{cmp}_{V,A} \ (f_1, f_2 \in \mathfrak{F}_{V,A}, f_1 \preceq f_2).$$

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:

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

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.

Proof. By Theorem 4 the lattice $([f_1, f_2], \cup, \cap)$ is distributive.

From the definition of $\mathsf{C}_{[f_1,f_2]}(f)$ it follows that for each $f \in [f_1, f_2]$

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

$$= graph(f) \cup (graph(f_2) \backslash graph(f) \cup graph(f_1)) =$$

$$= (graph(f) \cup graph(f_2) \backslash graph(f)) \cup graph(f_1) =$$

$$= graph(f_2) \cup graph(f_1) = graph(f_2),$$

i.e. $f \cup \mathsf{C}_{[f_1,f_2]}(f) = f_2$, and also,

$$graph(f) \cap graph\mathsf{C}_{[f_1,f_2]}(f) =$$

$$= graph(f) \cap (graph(f_2) \backslash graph(f)) \cup graph(f_1)) =$$

$$= (graph(f) \cap (graph(f_2) \backslash graph(f))) \cup (graph f \cap graph(f_1)) =$$

$$= \emptyset \cup graph(f_1) = graph(f_1),$$

i.e. $f \cap \mathsf{C}_{[f_1,f_2]}(f) = f_1$.

Since $f \cup \mathsf{C}_{[f_1,f_2]}(f) = f_2$ and $f \cap \mathsf{C}_{[f_1,f_2]}(f) = f_1$, the element $\mathsf{C}_{[f_1,f_2]}(f) \in [f_1, f_2]$ is a relative complement of the element $f \in [f_1, f_2]$ in the interval $[f_1, f_2]$.

By the definition, a distributive lattice with a relative complement 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 $\varphi|_{[f_1^{(1)},f_2^{(1)}]}$ is bijection of $[f_1^{(1)}, f_2^{(1)}]$ onto $[f_1^{(2)}, f_2^{(2)}]$ for which the identities

$$\varphi|_{[f_1^{(1)},f_2^{(1)}]}(f' \cup f'') = \varphi|_{[f_1^{(1)},f_2^{(1)}]}(f') \cup \varphi|_{[f_1^{(1)},f_2^{(1)}]}(f'')$$

and

$$\varphi|_{[f_1^{(1)},f_2^{(1)}]}(f' \cap f'') = \varphi|_{[f_1^{(1)},f_2^{(1)}]}(f') \cap \varphi|_{[f_1^{(1)},f_2^{(1)}]}(f'')$$

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* $[0_{V,A}, f_2 \backslash f_1]$.

Proof. Let $\varphi : \mathfrak{F}_{V,A} \to \mathfrak{F}_{V,A}$ be any mapping such that $\varphi(f) = f \backslash f_1$ holds for all $f \in [f_1, f_2]$.

Then $\varphi(f_1) = f_1 \backslash f_1 = 0_{V,A}$, $\varphi(f_2) = f_2 \backslash f_1$, $0_{V,A} \preceq \varphi(f) \preceq f_2 \backslash f_1$ for all $f \in [f_1, f_2]$, i.e. φ maps the interval $[f_1, f_2]$ onto the interval $[0_{V,A}, f_2 \backslash f_1]$.

If $f' \neq f''$ $(f', f'' \in [f_1, f_2])$, then

$$\varphi(f') = f_2 \backslash f' \neq f_2 \backslash f'' = \varphi(f'').$$

Thus, $\varphi|_{[f_1,f_2]}$ is a bijection from $[f_1, f_2]$ onto the interval $[0_{V,A}, f_2\backslash f_1]$.

Since $f_1 \preceq f'$ $f_1 \preceq f''$ for any elements $f', f'' \in [f_1, f_2]$, we have

$$\varphi(f' \cup f'') = (f' \cup f'')\backslash f_1 = f'\backslash f_1 \cup f''\backslash f_1 = \varphi(f') \cup \varphi(f'')$$

and

$$\varphi(f' \cap f'') = (f' \cap f'')\backslash f_1 = f'\backslash f_1 \cap f''\backslash f_1 = \varphi(f') \cap \varphi(f'').$$

Thus $\varphi|_{[f_1,f_2]}$ is an isomorphism from $[f_1, f_2]$ onto $[0_{V,A}, f_2\backslash f_1]$.

□

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

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

Proof. Let us fix any elements $f^{(1)}, f^{(2)} \in \mathfrak{F}^{(ttl)}_{V,A}$ $(f^{(1)} \neq f^{(2)})$ and consider maximal intervals $[0_{V,A}, f^{(1)}]$ and $[0_{V,A}, f^{(2)}]$. Let $g = f^{(2)}\backslash f^{(1)}$. Let us define a mapping $\varphi : \mathfrak{F}_{V,A} \to \mathfrak{F}_{V,A}$ as follows:

$$\varphi(f) = g|_{\mathrm{Dom} f} \triangleright f \ \ (f \in \mathfrak{F}_{V,A}).$$

From this it follows that

$$\varphi(0_{V,A}) = g|_{\mathrm{Dom} 0_{V,A}} \triangleright 0_{V,A} = g|_{\emptyset} \triangleright 0_{V,A} = 0_{V,A} \triangleright 0_{V,A} = 0_{V,A},$$

$$\varphi(f^{(1)}) = g|_{\mathrm{Dom} f^{(1)}} \triangleright f^{(1)} = g|_{V} \triangleright f^{(1)} = g \triangleright f^{(1)} = f^{(2)},$$

and also, $0_{V,A} \preceq \varphi(f) \preceq f^{(2)}$ for all $f \in [0_{V,A}, f^{(1)}]$, i.e. φ maps the interval $[0_{V,A}, f^{(1)}]$ onto the interval $[0_{V,A}, f^{(2)}]$.

Since for any elements $f', f'' \in [0_{V,A}, f^{(1)}]$ $(f' \neq f'')$ the inequality $\mathrm{Dom} f' \neq \mathrm{Dom} f''$ holds, at least one of the inequalities

$$\mathrm{Dom} g \cap \mathrm{Dom} f' \neq \mathrm{Dom} g \cap \mathrm{Dom} f''$$

or

$$\mathrm{Dom} f'\backslash \mathrm{Dom} g \neq \mathrm{Dom} f''\backslash \mathrm{Dom} g$$

holds. From this it follows that if $f' \neq f''$ ($f', f'' \in [0_{V,A}, f^{(1)}]$), then $\varphi(f') \neq \varphi(f'')$.

Thus $\varphi|_{[0_{V,A}, f^{(1)}]}$ is a bijection from the interval $[0_{V,A}, f^{(1)}]$ onto the interval $[0_{V,A}, f^{(2)}]$.

For each $f', f'' \in [0_{V,A}, f^{(1)}]$ we have

$$\varphi(f' \cup f'') = g|_{\mathrm{Dom}(f' \cup f'')} \triangleright (f' \cup f'') = g|_{\mathrm{Dom}(f' \cup f'')} \triangleright f' \cup g|_{\mathrm{Dom}(f' \cup f'')} \triangleright f'' =$$

$$= g|_{\mathrm{Dom} f'} \triangleright f' \cup g|_{\mathrm{Dom} f''} \triangleright f'' = \varphi(f') \cup \varphi(f'').$$

From the definition of φ and Proposition 3(vii) it follows that for each $f', f'' \in [0_{V,A}, f^{(1)}]$,

$$\varphi(f' \cap f'') = g|_{\mathrm{Dom}(f' \cap f'')} \triangleright (f' \cap f'') = g|_{\mathrm{Dom}(f' \cap f'')} \triangleright f' \cap g|_{\mathrm{Dom}(f' \cap f'')} \triangleright f'' =$$

$$= g|_{\mathrm{Dom} f'} \triangleright f' \cup g|_{\mathrm{Dom} f''} \triangleright f'' = \varphi(f') \cup \varphi(f'').$$

Thus $\varphi|_{[f_1, f_2]}$ is an isomorphism from the interval $[0_{V,A}, f^{(1)}]$ onto the interval $[0_{V,A}, f^{(2)}]$. $\square$

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. In this context investigation of algebras of programs over nominative data is essential.

In the given paper we restricted ourselves to basic (flat) nominative data called nominative sets. Nominative sets adequately represent

such commonly used data structures as arrays, records, and dictionaries. Hierarchical types of data can naturally represent a much larger set of data structures used in programming, including multidimensional arrays, lists, trees, algebraic data types, etc. The details of such representation are given in [16]. These types of nominative data induce sufficiently rich program algebras. We plan to investigate such algebras in future papers.

References

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

[2] R. Floyd. *Assigning meanings to programs.* In: Proceedings of the American Mathematical Society Symposia on Applied Mathematics. Volume 19, pp. 19–31, 1967.

[3] C.A.R. Hoare. *An axiomatic basis for computer programming.* Communications of the ACM, pp. 576–580, 1969.

[4] M.A. Jackson. *Principles of program design.* Academic Press, London, 1975.

[5] J. Backus. *Can programming be liberated from the von Neumann style? A functional style and its algebra of programs.* Communications of the ACM, Vol. 21, 1978, pp. 613–641.

[6] C.A.R. Hoare, J. Misra, G. Leavens, N. Shankar. *The verified software initiative: A manifesto.* ACM Computing Surveys. Volume 41, Issue 4, 2009.

[7] T. Hoare. *The verifying compiler: A grand challenge for computing research.* Gresham College, 18/03/2004, Barnard's Inn Hall, 2004. `http://www.cs.ox.ac.uk/files/6187/Grand.pdf`

[8] H.R. Nielson, F. Nielson. *Semantics with applications – a formal introduction.* Wiley professional computing. Wiley, 1992.

[9] D. Sannella, A. Tarlecki. *Foundations of Algebraic Specification and Formal Software Development.* Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2012.

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

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

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

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

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

[15] M. Nikitchenko, Ie. Ivanov. *Programming with nominative data.* In: Proceedings of CSE'2010 International Scientific Conference on Computer Science and Engineering, September 20–22, 2010, Kosice, Slovakia, pp. 30–39, 2010.

[16] V.G. Skobelev, M. Nikitchenko, Ie. Ivanov. *On algebraic properties of nominative data and functions.* Communications in Computer and Information Science (CCIS), Springer International Publishing, Volume 469, pp. 117–138, 2014.

[17] A. Voronkov. *EasyChair.* WWV 2010. 6th International Workshop on Automated Specification and Verification of Web Systems (Laura Kovacs and Temur Kutsia eds.). EasyChair Proceedings in Computing. Volume 18, 2013.
`http://www.easychair.org/publications/paper/EasyChair`

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

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

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

Volodymyr G. Skobelev, Ievgen Ivanov,
Mykola Nikitchenko

Received October 19, 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

Computer Science Journal of Moldova, vol.**23**, no.3(69), 2015

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

Alexander Lyaletski

Abstract

The paper is about the notions of admissibility and compatibility and their significance for deducibility in different sequent logics including first-order classical and intuitionistic ones both without and with equality and, possibly, with modal rules. Results on the coextensivity of the proposed sequent calculi with usual Gentzen and Kanger sequent calculi as well as with their equality and modal extensions are given.

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

1 Introduction

There is a great impact of methods originally developed for deduction in different logics on some branches of computer science. From the beginning it was realized that logical inference tools have strong influence on the development of such fields as automated theorem proving, knowledge management, data mining, etc. As a result, investigations in computer-made reasoning gave rise to the appearance of various methods for proof search in the classical first-order logic. Thus, Gentzen's sequent calculi [1] modified for their software implementation have found many applications. But in the case of the classical logic their practical usage as a logical engine 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 these methods as compared to sequent

calculi, which is mainly connected with different possible orders of the quantifier rule applications in sequent calculi while the 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 inferences are performed within the scope of the signature of an 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 comprehensive for a man. Besides, preliminary skolemization is not a valid operation for many non-classical logics including the intuitionistic one while many of such logics are widely used 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. For this step of deduction to be sound, certain restrictions are put on the substitution. A substitution, satisfying these restrictions, is said to be admissible. Here we show how Gentzen's notion of an admissible substitution can be modified so that computer-oriented sequent calculi can be finally obtained for both classical and non-classical logics. For simplicity, we give a complete description of our approach for the classical logic without equality and briefly discuss a way (utilizing additionally a so-called compatibility) to use it for the intuitionistic logic as well as for their equality and modal extensions.

We use modifications of the calculi LK and LJ without equality from [1] denoting them by mLK and mLJ respectively. Moreover, we convert mLK and mLJ in a certain way to logics with equality and/or 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. Note that in contrary to [1], the antecedents and succedents of all the sequents under consideration are assumed to be multisets. As usual, the inference search in any calculus is of the form of the so-called inference tree "growing" from bottom to top in accordance with the order of 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 that 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 $A[t/x]$ denotes 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. 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 LK will be established below, while here we notice that the quantifier rules should be applied to all the quantifiers occurring in A_1, A_2, and B. Therefore, the classical notion of an admissible substitution yields 90 $(= 6!/(2!{\cdot}2!{\cdot}2!))$ different orders of quantifier rule applications to $A_1, A_2 \to B$. It is clear that the resolution-type methods allow avoiding this redundant work.

3 Kanger's notion of admissibility

To optimize the procedure of applying the quantifier rules, in [3] S.Kanger suggested his Gentzen-type calculus, denoted here by K. In calculus K a "pattern" of an inference tree is first constructed with the help of special variables, the so called parameters and dummies. At some instants of time, 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-given quantifier rules and in a certain splitting (in K) of the process of the "pattern" construction into stages. The rules $(\forall : left)$ and $(\exists : right)$ of K are as follows:

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

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 the notion of a Kanger-admissible substitution, which is more efficient than of the Gentzen one. For example, in the above-given 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 do not allow achieving the efficiency comparable with that when the skolemization is made. It is due to the fact that, as in case of the Gentzen notion of admissibility, it is required to select a certain order of the quantifier rule applications when an initial 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 the quantifier structures of formulas of an initial sequent and a substitution under consideration, would there exists a desired sequence of 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* s is defined as a finite (maybe, empty) set of ordered pairs [5], every of which consists of a variable, say, x, and a term, say, t, and is written as t/x, where x is called a *variable* and t a *term* of

s. For a sequent tree D, by $D \cdot s$ denote 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 of $t\}$. The substitution s is said to be *admissible* for P if and only if (1) the variables of s are dummies and (2) there are no triples $\langle z_1, t_1, p_1\rangle, \ldots, \langle z_n, t_n, p_n\rangle \in T(P, s)$ 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 sequents consisting of formulas, except that in mLK, to every formula, a (possibly, empty) sequence of parameters and dummies is additionally assigned. Thus, the sequents of mLK consist of 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 each formula of an initial sequent (that is, a sequent to be proved).

The calculus mLK has the following rules:

Axioms:

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

Propositional rules:

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

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

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

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

Contraction rules:

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

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 a new parameter, p a sequence of parameters and dummies, Γ and Δ are arbitrary multisets of ordered 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* $\langle, A_1\rangle,\ldots,\langle, A_m\rangle \to \langle, B_1\rangle,\ldots,\langle, B_n\rangle$ in mLK (or its modifications).

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 an initial 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 converting all the leaves of D into axioms and being admissible for the set P of all sequences of parameters and dummies from D. Without loss of generality, it can be assumed 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 occurring in 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 can be generated, for example, by using the convolution algorithm from [4], applied to a list of all the sequences from P (in the convolution algorithm, parameters are treated as existence quantifiers and dummies as universal quantifiers). The property (i) of the sequence p and the definitions of the propositional and quantifier rules lead to the following observation:

When D is constructed, propositional, contraction, and quantifier rules are 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 is applied, the first variable (a parameter or dummy) of p is generated, when the second quantifier rule is applied, the second variable (a parameter or dummy) of p is generated, and so on.

Now it is possible to convert the tree D into a proof tree D' for the initial sequent $A_1, \ldots, A_m \to B_1, \ldots, B_n$ in calculus LK. To do this, it is enough to "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:

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

where $t/d \in s$ 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 prop-

erty (ii) is satisfied. Hence, the conditions of admissibility in Gentzen's (classical) sense are satisfied when the above-given rules $(\forall : left')$ and $(\exists : right')$ are replaced in D by the corresponding rules $(\forall : left)$ and $(\exists : right)$ of the calculus LK:

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

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

2) In all the 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 an inference tree in the calculus LK. Furthermore, by the construction of D' it follows that all its leaves are axioms of the calculus LK. Thus, D' is a proof tree for the initial sequent A_1, $\ldots$, $A_m \to B_1, \ldots, B_n$ in LK.

(<=) Let D' be a proof tree for an initial sequent A_1, $\ldots$, $A_m \to B_1, \ldots, B_n$ in the calculus LK. Convert D' into a proof tree D 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. For this purpose, "repeat" (from bottom to top) the 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 substituting d for those occurrences of the term t into "heirs" of the formula $A[t/x]$ being the result of a replaced rule application inserting t.

2) In all the other cases, the replacement of the rules of LK by the rules of mLK is evident. (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. $\square$

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

Now let us apply propositional rules to the latter sequent as long as they are applicable. As a result, we construct an inference tree, say, D. If we take 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 the following conclusions concerning s and D are valid: (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 Admissibility, compatibility, and intuitionistic deducibility

The intuitionistic calculus LJ is distinguished from LK by that the succedent of any sequent in LJ should contain no more than one for-

mula [1]. In this connection, it may seem that this restriction putting on mLK leads 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 $\neg\forall x P(x) \to \exists y \neg P(y)$. Obviously, it is deducible in LK while it *is not deducible* in LJ.

We can construct the following proof tree D in mLK for it:

$$\begin{array}{c}
\langle d, P(d)\rangle \to \langle z, P(z)\rangle \\
\hline
\langle d, P(d)\rangle \to \langle, \forall x P(x)\rangle \\
\hline
\langle, \neg\forall x P(x)\rangle, \langle d, P(d)\rangle \to \\
\hline
\langle, \neg\forall x P(x)\rangle \to \langle d, \neg P(d)\rangle \\
\hline
\langle, \neg\forall x P(x)\rangle \to \langle, \exists y \neg P(y)\rangle
\end{array},$$

where d is a dummy and z a parameter.

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

The succedent of any sequent in D contains only one formula, i.e. D satisfies the intuitionistic requirement to inference rules. Therefore, the usage of only the new admissibility is not enough for providing the "sound" deducibility in mLJ for the intuitionistic case.

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 formally defined 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 provides the soundness of inference search. For example, the above-given tree D for the sequent $\neg\forall x P(x) \to \exists y \neg P(y)$ *is not compatible* with the unique "reasonable" substitution $s = \{z/d\}$, which implies that $\neg\forall x P(x) \to \exists y \neg P(y)$ is not deducible in the calculus 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*.*

Pay your attention to the fact that Theorems 1 and 2 are distinguished by only the presence of the item (3) in Theorem 2.

7 Admissibility, compatibility, and deducibility in equality and modal extensions

Let $\mathrm{LK}^{\approx}$ and $\mathrm{LJ}^{\approx}$ be, respectively, the calculi LK and LJ, in which the Kanger equality rules from [3] are incorporated, where $\approx$ denotes the equality symbol.

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

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

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

As in [3], the introduced equality rules are applied in inference search in $\mathrm{mLK}^{\approx}$ and $\mathrm{mLJ}^{\approx}$ 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 be a substitution. Suppose that after subsequent applying 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 is produced, each leaf of which is only an axiom. 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 construction of modal calculi has a certain correlation with the papers [7] and [8], where necessary modal rules in a sequent form are simply added to Gentsen's calculi LK and LJ.

Doing the same for LK and LJ and $LK^{\approx}$ and $LJ^{\approx}$, we obtain 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 that can be added to LK, LJ, $LK^{\approx}$, and $LJ^{\approx}$, any such modal rule is considered to be of the following general 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 $\Phi_1, \ldots, \Phi_k$, $\Psi_1, \ldots, \Psi_r$ multisets of formulas (containing, possibly, modal operators). In particular, such approach makes possible to determine the calculus GK or GS4 from [8] based on using certain sequent rules for the standard modal operators $\Box$ and $\Diamond$.

Any modal rule of this form naturally determines 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 leads to the following result for modal extensions 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 [3]. 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 for all such modifications and extensions of K.*

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

8 Conclusion

The research presented in this paper demonstrates that the introduced notions of admissibility and compatibility lead to a good enough decision of the problem of quantifier handling in first-order logics. They can be easily built-in into the Gentzen calculi LK and LJ, which gives a good basis for constructing computer-oriented sequent calculi for classical and intuitionisticl logics as well as for their equality and modal extensions. Despite the questions of the machine implementation of such sequent calculi were not considered in the paper, note that 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, make a point that the Robinson unification algorithm combined with the new notion of admissibility (and compatibility) is suitable for generating such substitutions.

The suggested 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 the 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 Web-site "nevigal.org" as well as 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, France, 2010, pp. 411–426.

Alexander Lyaletski

Received October 27, 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

Computer Science Journal of Moldova, vol.**23**, no.3(69), 2015

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.

For an alphabet V, by V^* we denote the free monoid generated by V under the operation of concatenation, i.e., containing all possible strings over V. The *empty string* is denoted by λ.

In this paper we will not distinguish between a multiset, its string representation (having as many occurrences of every symbol as its multiplicity in the multiset, the order in the string being irrelevant), and a vector of multiplicities (assuming that the order of enumeration of symbols from V is fixed). We mention that $\prod$ represents the concatenation of an ordered list of strings, and if these strings represent multisets, this corresponds to their union.

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:*

- $p : (\texttt{ADD}\,(r)\,, q, s)$*, 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*.*
- $p : (\texttt{SUB}\,(r)\,, q, s)$*, 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* $p : (\texttt{ADD}\,(r)\,, q)$*.*

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, whereas *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.

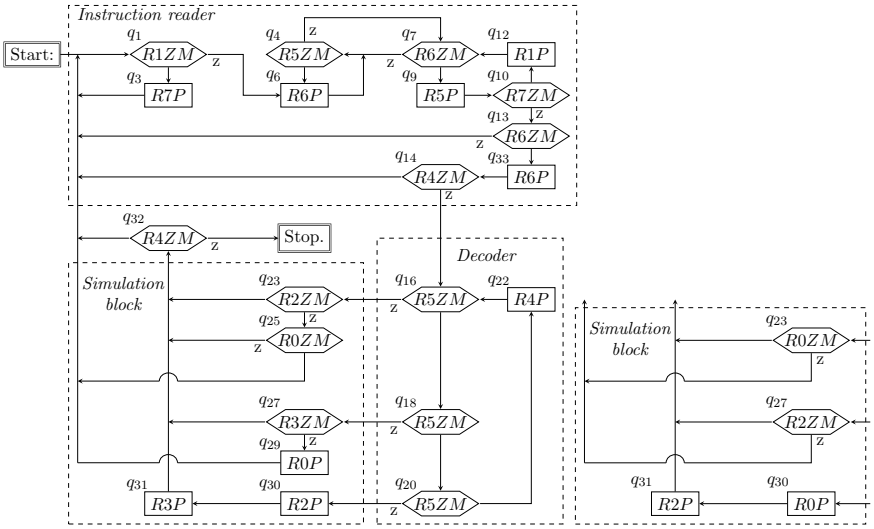

Figure 1. The strongly universal register machine U_{22} (left) and the simulation block of the weakly universal register machine U_{20} (right).

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 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 $j : (\texttt{SUB}(r), A_-(j)k, A_0(j)l)$ where $j, k, l \in B$ are instruction labels and $A_-(j), A_0(j)$ are (possibly empty) strings of increment commands (sub-instructions) $\texttt{ADD}(j')$. 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:

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

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:

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

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:*

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

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, $\mathtt{ADD}(r)$ is simulated by multiplying the value of the first register by p_r, and $\mathtt{SUB}(r)$ 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 $j : (\mathtt{ADD}(r), k)$ is simulated by the two generalized SUB-instructions $j : (\mathtt{SUB}(1), (\mathtt{ADD}(2))^{p_r}\ j, j')$ and $j' : (\mathtt{SUB}(2), \mathtt{ADD}(1)j', k)$.
– Instruction $j : (\mathtt{SUB}(r), k, l)$ is simulated by the $p_r + 2$ generalized SUB-instructions

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

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:

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

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 $h_8 : (\texttt{SUB}(1), h_8, h_9)$ 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.*

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

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

and R_1 contains the following rules:

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

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].

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:

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

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

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

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

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

and the rules from the set R are listed below:

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

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.

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 $\{d_{r,-}\,,\ d_{r,0} \mid 1 \le r \le m\}$ such that, for $1 \le r \le m$, $S(d_{r,0}) = 1$ 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.

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

$$
\begin{array}{rcl}
\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_\# \\
 & & 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,-}, \\
 & & 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, \\
 & & 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}), \\
 & & c_p q_{4'} \to c_p q_{10} d' e_S(\lambda) o_5, \\
 & & c_p q_{10} \to c_p q_{10} d' e_S(d_{6,-}\, d_{7,-}) o_1 o_5, \\
 & & c_p q_{10} \to c_p q_4 d' e_S(d_{7,-}\, d_{6,0}) o_1, \\
 & & 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}), \\
 & & c_p q_{10} \to c_p q_{18} d' e_S(d_{5,-}\, d_{6,-}\, d_{4,0}\, d_{7,0}), \\
 & & c_p q_{18} \to c_p q_{18} d' e_S(d_{5,-}\, d_{5,-}\, d_{5,-}) o_4, \\
 & & c_p q_{18} \to c_p q_1 d' e_S(d_{3,0}\, d_{5,0}) o_0, \\
 & & 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, \\
 & & c_p q_{18} \to c_p q_{32} d' e_S(d_{2,-}\, d_{5,-}\, d_{5,-}\, d_{5,0}) o_4, \\
 & & 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, \\
 & & c_p q_{18} \to c_p q_{32} d' e_S(d_{3,-}\, d_{5,0}), \\
 & & 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,-}), \\
 & & c_p q_{32} \to c_p q_{32} d' e_S(d_{0,-}\, d_{4,0}) o_8, \\
 & & 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}), \\
 & & c_p q_{32} \to c_p e_S(d_{0,0}\, d_{1,0}\, d_{4,0}\, d_{6,0})\}.
\end{array}
$$

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].

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 in-

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.

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	$98_{[3]}$			$89_{[3]}$		
$pcat_9$	$145_{[3]}$					
$\mathbf{pcat_8}$			$\mathbf{133}_{[17]+[4]}$	$120_{[4]}$		
$pcat_7$						$111_{[4]}$
$\mathbf{pcat_4}$		$\mathbf{611}_{+[4]}$	$\mathbf{476}_{+[4]}$		$\mathbf{608}_{+[4]}$	$\mathbf{473}_{+[4]}$
$\mathbf{cat_2}$		$\mathbf{611}_{+[4]}$	$\mathbf{476}_{+[4]}$		$\mathbf{608}_{+[4]}$	$\mathbf{473}_{+[4]}$
$\mathbf{pcat_3}$		$\mathbf{733}_{[17]+[4]}$	$\mathbf{571}_{[17]+[4]}$			
$pcat_2$					$726_{[4]}$	$564_{[4]}$

dicates 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.

Acknowledgements The work of the third author was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

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 Systems, 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.* In: M. Dinneen, Ed., Workshop on Membrane Computing, Auckland, 2015, 1–16.

[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 November 2, 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

Computer Science Journal of Moldova, vol.**23**, no.3(69), 2015

Extensionality, Proper Classes, and Quantum Non-Individuality

William J. Greenberg

Abstract

In this paper I address two questions: (1) What distinguishes proper classes from sets? (2) Are proper classes and quantum particles individuals?

Against the familiar response to (1) that proper classes are too big to be sets, I propose that it is not a difference in size that distinguishes such collections but a difference in individuation. The linchpin of my proposal and centerpiece of an NBG-like fragment of class and set theory ("NBG": von Neumann-Bernays-Gödel), is an Axiom of Restricted Extensionality according to which sets are individuated by their members but proper classes are not. This setting (I call it NBG$^-$) I show to be equi-consistent with its NBG counterpart.

I answer (2) by exhibiting a parallelism in NBG$^-$ between proper classes and quantum particles, the former unindividuated by their members and the latter unindividuated by their relational properties. Since both violate the (weak) principle of the identity of indiscernibles as well as the principle of reflexive identity, in NBG$^-$ neither proper classes nor quantum particles are individuals.

1 Three Principles

Modulo the deductive apparatus of First-Order Logic with Weak Identity[1], Russell's Paradox follows from three principles: *Unrestricted*

[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=.

Extensionality, *Restricted Comprehension*, and *Unrestricted Pairing*. Concerning the first of these Michael Potter writes:

> 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.

[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 1
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

(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*)

(D) can be avoided by replacing (B) with (B′), as in Zermelo Set Theory;

$$(\text{B}') \ \forall z \exists y \forall x(x \in y \leftrightarrow (x \in z \ \& \ Px)) \quad (\textit{Separation})$$

or by replacing (C) with (C') as in NBG*, a sub-theory of NBG;

$$(\text{C}') \ \forall w \forall u((\text{set } w \ \& \ \text{set } u) \rightarrow \exists y \forall x(x \in y \leftrightarrow (x = w \vee x = u))) \quad (\textit{Restricted Pairing})$$

or by replacing (A) with (A′), as in NBG$^-$: an NBG-like theory with *Restricted Extensionality* and *Unrestricted Pairing*:

(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)	$\exists y \forall x(x \in y \leftrightarrow (\text{set } x \ \& \ Px))$	(*Restricted Comprehension*)
(C)	$\forall w \forall u \exists y \forall x(x \in y \leftrightarrow x = w \vee x = u)$	(*Unrestricted Pairing*)

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).

[3] $\text{Prop } x \overset{\text{def}}{=} \neg(\text{set } x)$

T4: $\forall y(\forall x(x \in y \leftrightarrow (\text{set } x \ \& \ \neg(x \in x))) \rightarrow \text{prop } y)$

T5: $\forall y(\forall x(x \in y \leftrightarrow (\text{set } x \ \& \ \neg(x \in x))) \rightarrow \neg(y = y))$

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*)

[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* line 2
13. $\forall a\forall b\exists y\forall x(x \in y \leftrightarrow x = a \vee x = b)$	2, UG: Cancel *Show* 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* 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 1: "Set x" doesn't appear on the right-hand side of T8 or T9 because it is redundant.

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 by a non-self-identical, non-element. But such an entity violates model-theoretic restrictions 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*, and "I" in terms of "$\in$" in NBG$^-$; and then show that NBG* $\vdash$ NBG$^-$, and NBG$^-$ $\vdash$ 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)

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*:	$\text{set } x \overset{\text{def}}{=} \exists y(x \in y)$
	D2*:	$x = y \overset{\text{def}}{=} (x\mathrm{I}y \ \& \ \text{set } x \ \& \ \text{set } y)$

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)	$\mathrm{D1}^-$:	$\text{set } x \overset{\text{def}}{=} \exists y(x \in y)$
	$\mathrm{D2}^-$:	$x\mathrm{I}y \overset{\text{def}}{=} \forall z(z \in x \leftrightarrow z \in y)$

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. $x = y \overset{\text{def}}{=} (x\mathrm{I}y \ \& \ \text{set } x \ \& \ \text{set } y)$	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 $\forall x \forall y(x = y \to \forall z(z \in x \leftrightarrow z \in y))$	
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. $\forall x \forall y(x\text{I}y \to \forall z(z \in x \leftrightarrow z \in y))$	3*
7. $\forall z(z \in x \leftrightarrow z \in y)$	5, 6: Cancel *Show* line 3
8. $\forall x \forall y(x\text{I}y \to \forall z(z \in x \leftrightarrow z \in y))$	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 $\forall x \forall y(\forall z(z \in x \leftrightarrow z \in y) \to x\text{I}y)$	
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. $\forall x \forall y(\forall z(z \in x \leftrightarrow z \in y) \to x\text{I}y)$	2, 3: Cancel *Show* line 1

Proof of 3*:

1. Show $\forall x \forall y(x\text{I}y \to \forall \text{z}(\text{z} \in \text{x} \leftrightarrow \text{z} \in \text{y}))$	
2. $x\text{I}y \stackrel{\text{def}}{=} \forall \text{z}(\text{z} \in \text{x} \leftrightarrow \text{z} \in \text{y})$	D2$^-$
3. $\forall x \forall y(x\text{I}y \to \forall \text{z}(\text{z} \in \text{x} \leftrightarrow \text{z} \in \text{y}))$	2: Cancel *Show* line 1

Proof of 5*:

1. Show $\forall w \forall u((\text{set } w \text{ \& set } u) \to \exists y \forall x(x \in y \leftrightarrow (x\text{Iw} \vee \text{xIu})))$	
2. set w & set u	Assume
3. Show $\exists y \forall x(x \in y \leftrightarrow (x\text{Iw} \vee \text{xIu}))$	
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. $(x = w \vee x = u) \to \text{set } x$	1^-
7. $x \in y \to \text{set } x$	$D2^-$
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. $\exists y \forall x(x \in y \leftrightarrow x\text{Iw} \vee \text{xIu})$	11, D2$^-$: Cancel *Show* line 3
13. $\forall w \forall u((\text{set } w \text{ \& set } u) \to \exists y \forall x(x \in y \leftrightarrow (x\text{Iw} \vee \text{xIu})))$	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,

$$\text{T13: } \forall y(\forall x\neg(x \in y) \rightarrow \text{set } y) \quad (\textit{Empty Set})$$

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: $\forall x \forall y(\forall z(z \in x \leftrightarrow z \in y) \rightarrow (\text{set } x \leftrightarrow \text{set } y))$ (*Equi-Equi*)[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

Here is a set-theoretic gloss on the weak version of Leibniz's principle of the Identity of Indiscernibles (*PII*):

$\forall x \forall y(\forall z(x \in z \leftrightarrow y \in z) \rightarrow x = y)$ (*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 and $\neg(x = x)$

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], 276]

Whereof they cannot speak, thereof must Quasi-Set theories remain silent. But by remaining silent about the distinctness of indiscernible elementary particles, Quasi-Set theories dissimulate a relation whose trivial proof does nothing to diminish the bearing of the distinctness of indiscernibles on the Principle of Reflexive Identity (*PRI*).

For equi-propertied x and y, suppose $\neg(x = y)$. Because x lacks identity-with-y and x and y share their properties, y lacks identity-with-y and x identity-with-x. Hence for equi-propertied x and y: $\neg(x = y) \rightarrow (\neg(x = x)\ \&\ \neg(y = y))$. So distinct quantum particles with identical relational properties contravene *PRI* as well as unrestricted *PII*.

Except in the land of quasi-sets – where to defend ZFU from *Logic*, m-particles representing the elementary basic entities of quantum physics are not allowed to co-occur with the sign for identity.[12]

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*)

[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 wellformed formula for indistinguishable objects (see French & Krause [1999] and Krause [1992])". ([13], 106)

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: $\forall w \forall u \exists y \forall x(x \in y \leftrightarrow x = w \vee x = u)$	*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$^-$.

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*

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 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.

[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.

[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 $x \stackrel{\text{def}}{=} \exists y(x \in y \;\&\; \forall z(z \in y \to z = x))$ $\qquad (\stackrel{\text{def}}{=} \mathit{Set}\ )$

From ($\stackrel{\text{def}}{=} \mathit{Set}$) 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:

$x = x \leftrightarrow \exists y(x \in y \;\&\; \forall z(z \in y \to z = x))$ $\qquad (\mathit{Ind})$

Hence $\neg(x = x)$ if, and only if, x is not a member of any unit class.

$\neg(x = x) \leftrightarrow \forall y(\neg(x \in y) \vee \exists z(z \in y \;\&\; \neg(z = x)))$ $\qquad (\mathit{Ind})$

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 such that it contains an element that is not identical with x (think quantum particles). In both cases, x can be said to *lack individuality*.

[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*. 2nd 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

Received November 2, 2015

USA
E–mail: wgreenb@gmail.com

Computer Science Journal of Moldova, vol.**23**, no.3(69), 2015

Edge detection in digital images using Ant Colony Optimization

Marjan Kuchaki Rafsanjani, Zahra Asghari Varzaneh

Abstract

Ant Colony Optimization (ACO) is an optimization algorithm inspired by the behavior of real ant colonies to approximate the solutions of difficult optimization problems. In this paper, ACO is introduced to tackle the image edge detection problem. The proposed approach is based on the distribution of ants on an image; ants try to find possible edges by using a state transition function. Experimental results show that the proposed method compared to standard edge detectors is less sensitive to Gaussian noise and gives finer details and thinner edges when compared to earlier ant-based approaches.

Keywords: Ant Colony Optimization (ACO), Digital image processing, Edge detection, Noisy images.

1 Introduction

Edge detection is by far the most common approach for detecting meaningful discontinuities in gray level. It is an important problem in pattern recognition, computer vision and image processing. Conventional image edge detection algorithms usually perform a linear filtering operation (or with a smoothing pre-processing operation to remove noise from the image) on the image [1], such as Sobel, Prewitt [2] and Canny operators [3].

Ant Colony Optimization (ACO) is an optimization algorithm inspired by the behavior of real ant colonies [4, 5]. Ants deposit pheromone on the ground to mark their favorable paths, which can be followed by the ants of the colony. The first ACO algorithm, called

the ant system, was proposed by Dorigo et al. [4]. Since then, a number of ACO algorithms have been developed, such as the Max-Min ant system [6]. ACO has been widely applied in various problems [7, 8, 9, 10, 11]. Besides, ACO algorithms has been used to solve many complex problems successfully such as quadratic assignment problem [12], data clustering [13], image retrieval [14], too used to image thresholding [15], and image segmentation [16, 17].

Zhuang [18] proposed to utilize the perceptual graph to represent the relationship among neighboring image pixels, then use the ant colony system to build up the perceptual graph. Nezamabadi-Pour et al. [19] proposed to use the ant system to detect edges from images by formulating the image as a directed graph. Lu and Chen [20] proposed to use the ACO technique as a post-processing to compensate broken edges, which are usually incurred in the conventional image edge detection algorithms. Alikhani et al. [21] use a Fuzzy Inference System (FIS) with 4 simple rules to identify the probable edge pixels in 4 main directions, then the ACO is applied for assigning a higher pheromone value for the probable edge pixels. Finally, by using an intelligent thresholding technique which is provided by training a neural network, the edges from the final pheromone matrix are extracted. Davoodianidaliki et al. [22] proposed usage of traditional edge detectors for initial pheromone and distribution matrixes that previously were equal and random. Koner and Acharyya [23] have applied the variants for detection of edges in binary images. Contreras et al. [24] propose an approach based on a paradigm that arises from artificial life; more specifically ant colonies foraging behavior. Ari et al. [25] proposed a novel algorithm for image edge detection using ant colony optimization and Fisher ratio (Fratio)-based techniques. Ming and Xianghong [26] used ant colony system algorithm (ACSA) to detect the edge of gray scale images. The novelty of the proposed method is that the artificial ants used for detecting the edges of images have global memory capacity. Method proposed by Agrawal et al. [27] gives a pheromone matrix and memory stored positions that are followed by leading ant. The memory based positions are stored on the basis of intensity values with reference with a threshold value. Tian et al. [28]

proposed to establish a pheromone matrix that represents the edge presented at each pixel position of the image, according to the movements of ants on the image.

In this paper, we propose an improved ant-based edge detector that provides finer details and thinner edges on both noisy and clean images. This method uses pheromone information to detect the edges of image.

The remainder of the paper is organized as follows; the next section describes Ant Colony Optimization. In Section 3, we discuss Ant-Based our edge detection approach in details. Experimental results and analysis are presented in Section 4. Finally, Section 5 concludes this paper.

2 Ant Colony Optimization (ACO)

In the natural world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail; returning and reinforcing it if they eventually find food. Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Pheromone evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained.

Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads to all the ants' following a single path. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the graph representing the problem to solve. The ant colony optimization algorithm is a probabilistic

technique for solving computational problems which can be reduced to finding better paths through graphs. This algorithm is a member of the ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheuristic optimizations. Initially proposed by Marco Dorigo in 1992 in his PhD thesis [4]; the first algorithm was aiming to search for an optimal path in a graph, based on the behavior of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behavior of ants.

The goal of this article is to introduce an ant-based algorithm for edge detection.

3 The proposed Ant-Based Approach

In this approach, the value of visibility is determined using the maximum variation of gray level of the image intensity. Edge pixels are expected to have a greater value of visibility. Therefore, the ants' movements are driven by the local variation of the image intensity values. That is, the ants prefer to move towards positions with larger variations [12].

The proposed approach works as follows:

Step1: At first, k ants are placed on the randomly chosen nodes (pixel position) on an image I with a size of $M1 \times M2$. Therefore, one ant is assigned to each pixel position (called a node) of the image. The proposed approach sets the initial value of each component of the pheromone matrix $\tau^{(0)}$ to be a constant τ_{init}.

Step2: At the n-th construction-step, each ant probabilistically selects a new neighbor pixel to visit according to Eq. (1). The probability of displacing k-th ant from node (l,m) to its neighboring node (i,j) is determined by:

$$P^{(n)}_{(l,m),(i,j)} = \frac{(T^{(n-1)}_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}{\sum_{(s,q)\in\Omega(l,m)}(T^{(n-1)}_{s,q})^{\alpha}(\eta_{s,q})^{\beta}}, \tag{1}$$

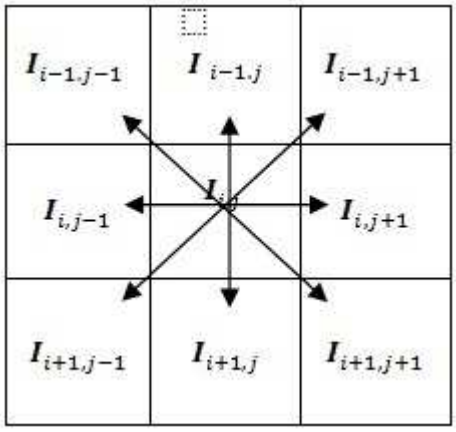

Figure 1. A local configuration at the pixel position $I_{i,j}$ for computing the variation $V_c(I_{i,j})$ defined in (3).

where $\Omega(l,m)$ is the neighborhood nodes of the node (l,m); $\eta_{i,j}$ and $T_{i,j}^{(n-1)}$ are the heuristic information which belongs to pixel (i,j) and the pheromone intensity of the pixel (i,j), respectively; and, the parameters α and β control the relative importance of the pheromone matrix versus the heuristic information $\eta_{i,j}$ used in [28], which is given by:

$$\eta_{i,j} = \frac{1}{z} V_c(I_{i,j}), \tag{2}$$

where $Z = \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} V_c(I_{i,j})$ which is a normalization factor, $I_{i,j}$ is the intensity value of the pixel at the position (i,j) of the image I and function $V_c(I_{i,j})$ is a function of a local group of pixels c, and its value depends on the variation of image's intensity values on the clique c (as shown in Figure 1). The function $V_c(I_{i,j})$ is determined by:

$$\begin{aligned} V_c(I_{i,j}) = f(&|I_{i-1,j-1} - I_{i+1,j+1}| + |I_{i-1,j} - I_{i+1,j}| + \\ &|I_{i-1,j+1} - I_{i+1,j-1}| + |I_{i,j-1} - I_{i,j+1}|). \end{aligned} \tag{3}$$

To determine the function f (.), the following four functions are considered [8]; they are mathematically expressed as follows:

$$f(x) = \lambda x \qquad x \geq 0; \tag{4}$$

$$f(x) = \lambda x^2 \qquad x \geq 0; \tag{5}$$

$$f(x) = \begin{cases} sin(\frac{\pi x}{2\lambda}) & 0 \leq x \leq \lambda \\ 0 & else \end{cases}; \tag{6}$$

$$f(x) = \begin{cases} sin(\frac{\pi x \ sin(\frac{\pi x}{\lambda})}{\lambda}) & 0 \leq x \leq \lambda \\ 0 & else \end{cases}. \tag{7}$$

We select f (.) which is defined by (8), because this function shows better results.

$$f(x) = \begin{cases} sin(\frac{\pi x}{2\lambda}) & 0 \leq x \leq \lambda \\ 0 & else \end{cases}. \tag{8}$$

The parameter λ determines the function's shape.

Step3: After every step, the pheromone values are updated after the movement of each ant within each construction-step according to:

$$\tau_{i,j}^{(n-1)} \leftarrow \begin{cases} (1-\rho).\tau_{i,j}^{(n-1)} + \rho.\Delta_{i,j}^{(k)} & if\ (i,j)\ is\ visited\ by\ the\ current \\ & k-th\ ant; \\ \tau_{i,j}^{(n-1)} & Otherwise \end{cases}, \tag{9}$$

where ρ is evaporation rate, it controls the degree of the updating of $\tau_{i,j}^{(n-1)}$; $\Delta_{i,j}^{(k)}$ is determined by the heuristic matrix; that is, $\Delta_{i,j}^{(k)} = \eta_{i,j}$. Secondly, after the movement of all ants, the pheromone matrix is updated as:

$$\tau^{(n)} = (1-\Psi).\tau^{(n-1)} + \Psi.\tau^{(0)}, \tag{10}$$

where Ψ is the pheromone decay coefficient, and $\tau^{(0)}$ is the initial value of the pheromone. Steps 2 and 3 iteratively run for N iterations. Finally, the pheromone matrix $\tau^{(n)}$ can be obtained to represent the saliency of the image.

Step4: Finally, a binary decision is made at each pixel location to determine whether it is edge or not, by applying a threshold T on the final pheromone matrix $\tau^{(N)}$. In this paper, we use thresholding based on the method developed in [1] as follows:

1. Select an initial estimate for T (average of the values for the points).
2. Produce two groups of values: G_1 consisting of all values $> T$ and G_2 consisting of values $< T$.
3. Compute the average values μ_1 and μ_2 for the values in G_1 and G_2.
4. Compute a new threshold value: $T = \frac{\mu_1+\mu_2}{2}$.
5. Repeat steps 2 through 4 until the difference in T in successive iterations is smaller than a predefined parameter ε.

4 Experimental Results

Experiments were conducted to demonstrate the performance of the proposed approach using two test images, Camera and House, which are shown in Figure 2.

4.1 Parameters Setting

Suitable algorithm parameters are determined based on trial and error. The parameters of the proposed approach were experimentally set as follows:

K is the number of ants. It could be chosen proportionally to the root of pixel numbers $M1 \times M2$. Total number of ant's movement-steps within each construction-step and total number of construction-steps

(a) (b)

Figure 2. Test images used in this paper: (a) Camera (256×256); (b) House (600×600).

are selected to be L= 50 and N= 4, respectively. τ_{init}, the initial value of each component of the pheromone matrix is set to be 0.0001; α and β control the relative importance of intensity of pheromone versus the heuristic information, they are set to be $\alpha = 6$ and $\beta = 0.001$, respectively [4]. The permissible ant's movement range at the position (l,m) could be either the 4-connectivity neighborhood or the 8-connectivity neighborhood. It is selected to be 8-connectivity neighborhood. λ is the adjusting factor of the functions, it is set to be 10. Evaporation rate and the pheromone decay coefficient are set to be $\rho = 0.1$ and $\Psi = 0.005$, respectively. Parameter ε is set to be 0.01.

4.2 Experimental Results and Discussions

Experimental results are provided to compare the proposed approach with Tian et al.'s edge detection method [28]. Figures 3 and 4 show the proposed approach that always outperforms Tian et al.'s method, in terms of visual quality of the extracted edge information. Therefore

Figure 3. Results of edge detectors for Cameraman image. (a) The original image; (b) Tian et al.'s edge detection algorithm [28]; (c) The proposed ACO-based image algorithm.

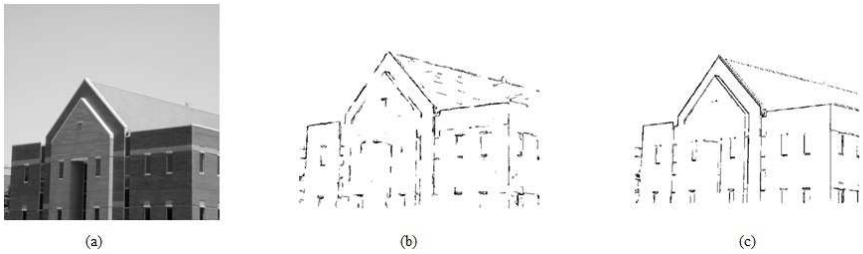

Figure 4. Results of edge detectors for House image. (a) The original image; (b) Tian et al.'s edge detection algorithm [28]; (c) The proposed ACO-based image algorithm.

the determination of parameters is critical to the performance of the proposed approach. Figures 5 and 7 show the results of Canny, Sobel, Log, Roberts and the proposed ACO-based edge detectors respectively on clear images. Furthermore we added Gaussian noise (see Figures 6 and 8) to test images. As we can see in the figures, our proposed method gives better results than the others, both in clean and noisy images.

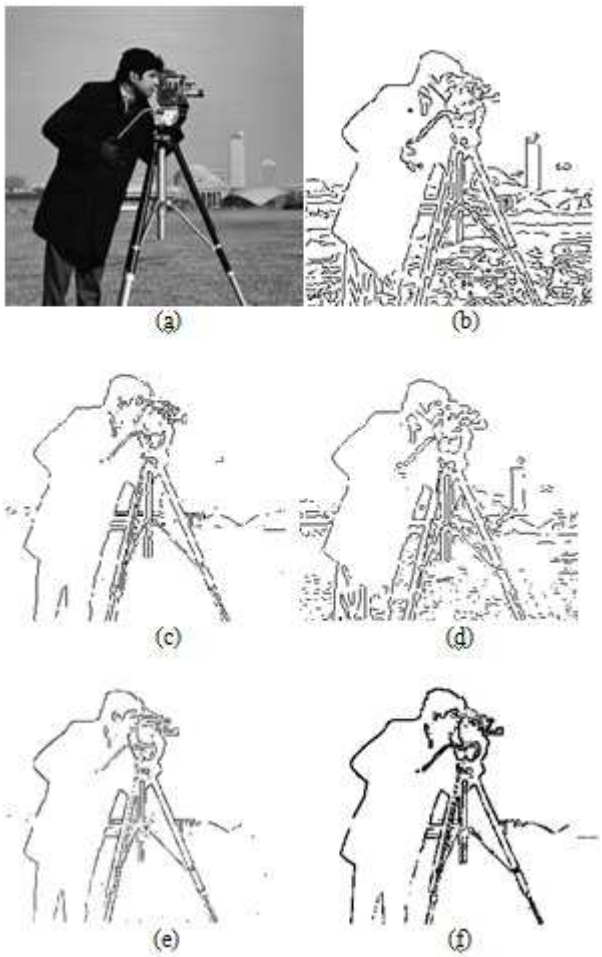

Figure 5. Comparison of ant-based edge detection algorithms. (a) The original image; (b) Canny edge detector; (c) Sobel edge detector; (d) Log edge detector; (e) Roberts edge detector (f) The proposed ACO-based edge detector.

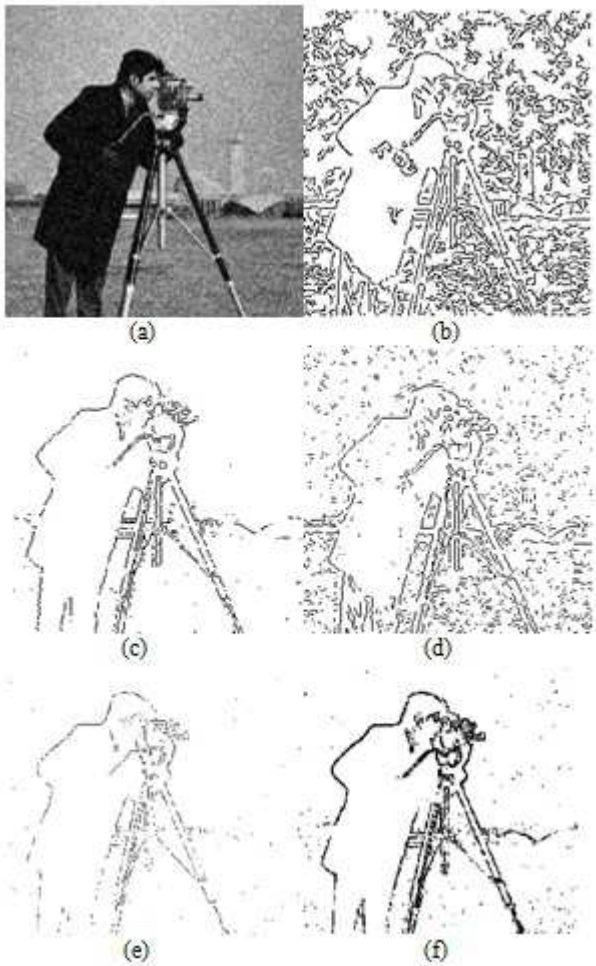

Figure 6. Comparison of ant-based edge detection algorithms. (a) The noisy image; (b) Canny edge detector; (c) Sobel edge detector; (d) Log edge detector; (e) Roberts edge detector (f) The proposed ACO-based edge detector.

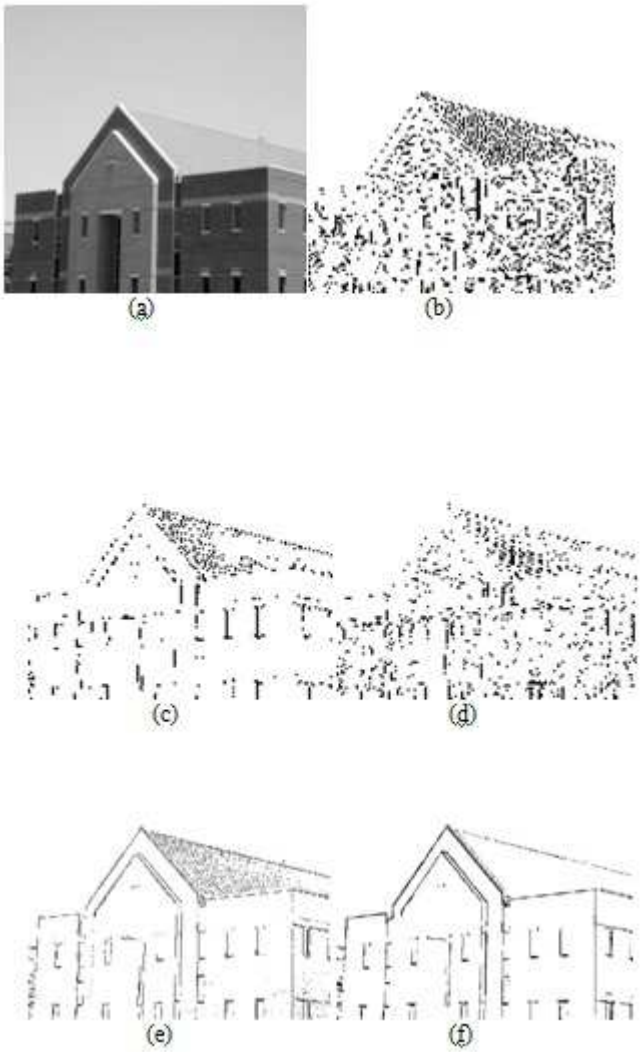

Figure 7. Comparison of ant-based edge detection algorithms. (a) The original image; (b) Canny edge detector; (c) Sobel edge detector; (d) Log edge detector; (e) Roberts edge detector (f) The proposed ACO-based edge detector.

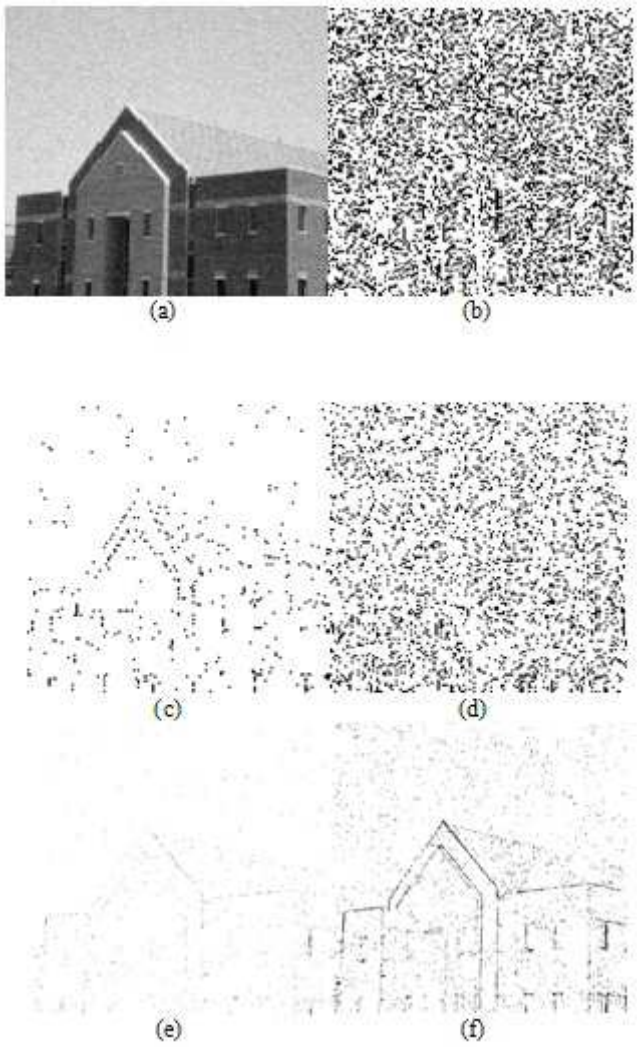

Figure 8. Comparison of ant-based edge detection algorithms. (a) The noisy image; (b) Canny edge detector; (c) Sobel edge detector; (d) Log edge detector; (e) Roberts edge detector (f) The proposed ACO-based edge detector.

5 Conclusions

This paper introduces an efficient ant-based edge detector that gives satisfactory results both in clean and noisy images. This approach uses a pheromone matrix that represents the edge presented at each pixel position of the image, according to the movements of ants on the image and updates the pheromone matrix using the initial pheromone value [28]. Suitable values of the algorithm parameters were determined through empirical studies. When we compare our proposed edge detector with other ant-based approaches, our edge detector gives finer details and thinner edges. Experimental results show that the proposed method compared to standard edge detectors is less sensitive to Gaussian noise.

References

[1] R. C. Gonzalez, R. E. Woods. *Digital image processing*, Prentice Hall, Upper Saddle River, 2002.

[2] A. C. Bovik. *Handbook of image and video processing*, Acedemic Press, New York, 1998.

[3] J. Canny. *A computational approach to edge detection*, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol.8 (1986), pp. 679–698.

[4] M. Dorigo. *Optimization, learning, and natural algorithms*, PhD Thesis, Dip Electronica e Informazione, Politeccnico di Milano, Italy, 1992.

[5] M. Dorigo, M. Birattari, S.Thomas. *Ant Colony Optimization*, Published by: IRIDIA, 2006.

[6] T. Stutzle, H. Holger H. *Max-Min ant system*, Future Generation Computer Systems, vol. 16 (2000), pp. 889–914.

[7] O. Cordon, F. Herrera, T. Stutzle. *Special Issue on Ant Colony Optimization: Models and Applications*, Mathware and Soft Computing, vol. 9 (2002).

[8] M. Dorigo, G.D. Caro, T. Stutzle. *Special Issue on Ant Algorithms*, Future Generation Computer Systems, vol. 16 (2000).

[9] M. Dorigo, L. M. Gambardella, M. Middendorf, T. Stutzle. *Special Issue on Ant Colony Optimization*, proceeding of the IEEE Transactions on Evolutionary Computation, vol. 6 (2002).

[10] S. Ouadfel, M. Batouche. *Ant colony system with local search for Markov random field image segmentation*, proceeding of the IEEE Int. Conf. on Image Processing, 133–136, 2003.

[11] A. T. Ghanbarian, E. Kabir, N. M. Charkari. *Color reduction based on ant colony*, Pattern Recognition Letters, vol. 28 (2007), pp. 1383–1390.

[12] T. Stutzle, M. Dorigo. *ACO algorithms for the quadratic assignment problem*, New ideas in optimization, McGraw-Hill, NewYork. pp. 33–50, 1999.

[13] B. Wu, Z. Shi. *A clustering algorithm based on swarm intelligence*, proceeding of the International Conferences Info-tech and Infonet, vol. 3 (2001), pp. 58–66.

[14] V. Ramos, F. Muge, P. Pina. *Self organized data and image retrieval as a consequence of inter-dynamic synergistic relationships in artificial ant colonies*, proceeding of the 2nd International Conference on Hybrid Intelligent Systems, vol. 87 (2002), pp. 500–509.

[15] A. R. Malisia, H. R. Tizhoosh. *Image thresholding using ant colony optimization*, proceeding of the Canadian Conference on Computer and Robot Vision, Quebec, Canada, pp. 26–26, 2006.

[16] W. Tao, H. Jin, L. Liu. *Object segmentation using ant colony optimization algorithm and fuzzy entropy*, Pattern Recognition Letters, vol. 28, no. 7 (2007), pp. 788–796.

[17] X. Zhuang. *Image segmentation with ant swarm - A case study of digital signal processing with biological mechanism of intelligence*, Proceeding of 11th Digital Signal processing workshop, pp. 143–146, 2004.

[18] X. Zhuang. *Edge feature extraction in digital images with the ant colony system*, proceeding of the IEEE Conference on Computational Intelligence for Measurement Systems and Applications, pp. 133–136, 2004.

[19] H. Nezamabadi-Pour, S. Saryazdi, E. Rashedi. *Edge detection using ant algorithms*, Soft Computing, vol. 10 (2006), pp. 623–628.

[20] D.S. Lu, C.C. Chen. *Edge detection improvement by ant colony optimization*, Pattern Recognition Letters, vol. 29 (2008), pp. 416–425.

[21] H.Reza-Alikhani, A. Naghsh, R.Jalali-Varnamkhasti. *Edge detection of digital images using a conducted ant colony optimization and intelligent thresholding*, 1st Iranian Conference on Pattern Recognition and Image Analysis, PRIA 2013, art. no. 6528432,2013.

[22] M. Davoodianidaliki, A. Abedini, M. Shankayi. *Adaptive edge detection using adjusted ant colony optimization*, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, 40 (1W3), pp. 123–126, 2013.

[23] S. Koner, S. Acharyya. *Ant colony optimization variants in image edge detection*, International Conference on Communication and Signal Processing, ICCSP 2014 - Proceedings, art. no. 6950034, pp. 1228–1232, 2014.

[24] R. Contreras, M.A. Pinninghoff, J. Ortega. *Using ant colony optimization for edge detection in gray scale images*, Lecture Notes

in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7930 LNCS (PART 1), pp. 323–331, 2013.

[25] S. Ari, D.K. Ghosh, P.K. Mohanty. *Edge detection using ACO and F ratio*, Signal, Image and Video Processing, vol. 8 no. 4 (2014), pp. 625–634.

[26] M. Shao, X. Xu. *Edge detection using ant colony system algorithm*, Proceedings of SPIE - The International Society for Optical Engineering, 8334, art. no. 83341X, 2012.

[27] P. Agrawal, S. Kaur, H. Kaur, A.Dhiman. *Analysis and synthesis of an ant colony optimization technique for image edge detection*, Proceedings: Turing 100 - International Conference on Computing Sciences, ICCS 2012, art. no. 6391659, pp. 127–131, 2012.

[28] J. Tian, W. Yu, L. Chen, L. Ma. *Image Edge Detection Using Variation-Adaptive Ant Colony Optimization*, Transactions on CCI V, pp. 27–40, 2011.

Marjan Kuchaki Rafsanjani, Zahra Asghari Varzaneh, Received March 29, 2015
Revised May 22, 2015

Marjan Kuchaki Rafsanjani
Department of Computer Science,
Faculty of Mathematics and Computer,
Shahid Bahonar University of Kerman,
Kerman, Iran
E–mail: kuchaki@uk.ac.ir

Zahra Asghari Varzaneh
Department of Computer Science,
Faculty of Mathematics and Computer,
Shahid Bahonar University of Kerman,
Kerman, Iran
E–mail: asghari_za@yahoo.com

Computer Science Journal of Moldova, vol.**23**, no.3(69), 2015

Probability on groups and an application to cryptography *

Sidoine Djimnaibeye, Daniel Tieudjo†, Norbert Youmbi

Abstract

In her thesis, Mosina introduced the concept of *mean-set of random (graph-) group-variables* and generalized Strong Law of Large Numbers (SLLN) to (graphs) groups, which she used for cryptanalysis of authentication schemes. This attack called the *mean-set attack* is presented here. It allows to break the Sibert authentication scheme on braid groups without solving the underlined difficult problem. We propose an amelioration to this attack and its implementation on the platform CRAG. We carry some experiments and we present the results. These results are discussed and they confirm those obtained by Mosina and Ushakov with a considerable gain of time.

Keywords: Braid group, Authentication protocol, Probability on groups, Mean-set attack, CRAG.

2010 Mathematics Subject Classification: primary 20F36, 60B15, secondary 14G50.

1 Introduction

During these last years, several cryptosystems among which the authentication schemes based on difficult problems in braid groups were proposed. Indeed, Sibert and *al.* [21] presented authentication schemes based on the conjugacy problem, the Diffie-Helmann-type conjugacy problem, the root problem; in [3], Dehornoy designed an authentication scheme using the shifted conjugacy problem; Lal and Chaturvedi

*This work was completed at Saint Francis University (Loretto, PA, USA) during the second author's visit thanks to the Fulbright Program. Saint Francis University and the Fulbright Scholar Program are gratefully acknowledged.

proposed in [8] two authentication schemes presumably based on the difficulty of the root problem; Shpilrain and Ushakov also offered in [20] an authentication scheme whose security is based on the hardness of the twisted conjugacy search problem, etc. The security of these authentication schemes relies on the difficulty to solve the underlined algorithmic problems. So, the robustness of these schemes is ensured by their resistance to the known cryptanalysis methods. Several methods to attack the authentication schemes were suggested in the literature [5, 6, 9, 10, 11, 23]. Mostly, these methods try to solve the difficult problem used to design the scheme.

In 2009 in her thesis, Natalia Mosina presented a new probabilistic approach to prove the vulnerability of the authentication protocols on the braid groups [14, 15, 16] without solving the underlying problem. So, given a group G with a probability measure induced by random G-variables, Mosina defined the mean-set of random G-elements. She stated and proved the Strong Law of Large Numbers (SLLN) on G, and gave an algorithm to compute the mean-set of a sequence of independent and identically distributed (i.i.d.) random G-variables. Using these tools, she developed an approach called *mean-set attack* that breaks the Sibert and *al.* authentication scheme, without solving the difficult problem used to design the protocol. She then implemented the attack and carried some experiments on the n-string braid group B_n with the software package CRAG. In her approach, Mosina considered the relative frequency as the probability distribution on the group B_n. However in [21], Sibert and *al.* suggested the use of the uniform law to generate the braids and the bits in the authentication protocol.

In this work, we present the Mosina's probabilistic approach and a restricted form which uses the uniform law. We derive a simplified mean-set attack algorithm that we implement on CRAG. We perform a series of experiments and discuss the results obtained. We see that they confirm those obtained by Mosina with a considerable gain of time.

2 Probability on groups

2.1 Mean set of a random G-variable

Let $G = \langle X \rangle$ be the group generated by a non empty set X. Let $C_G(X)$ be the Cayley graph associated to G. Let $(\Omega, \mathcal{F}, P)$ be a probability space and $\xi : \Omega \to G$ a random G-variable.

- A *probability distribution* is a function $\mu : G \to [0,1]$ on ξ such that:

$$\mu(g) = \mu_\xi(g) = P(\{\omega \in \Omega \mid \xi(\omega) = g\}, g \in G);$$

- The *weight function* is the function $M_\xi : G \to \mathbb{R}$ defined by

$$M_\xi(g) = \sum_{s \in G} d^2(g,s)\mu(s),$$

  where $d(g,s)$ is the distance between g and s in the Cayley graph $C_G(X)$ of G.

- The domain $domain(M)$ of the weight function M is defined by:

$$domain(M) = \left\{ g \in G \mid \sum_{s \in G} d^2(g,s)\mu(s) < \infty \right\}.$$

The weight function M_ξ is *totally defined* if for all vertices $g \in G$, $M_\xi(g) < \infty$ i.e. $domain(M) = G$.

Definition 2.1. *Let ξ be a random G-variable such that $M_\xi(\cdot)$ is totally defined. The set $\mathbb{E}(\xi)$ of vertices $g \in G$ having the smallest value of M_ξ i.e.*

$$\mathbb{E}(\xi) = \{g \in G : M_\xi(g) \leq M_\xi(u), \forall u \in G\}$$

is called mean-set of ξ.

Since $d(a,b) = d(ga, gb)$ for all a, b, $g \in G$ we have:

Proposition 2.1 (Shift Property)**.** Let $G = \langle X \rangle$ be the group generated by a non empty set X and let $g \in G$. Suppose $(\Omega, \mathcal{F}, P)$ is a probability space and let $\xi : \Omega \to G$ be a random G-variable on Ω. Then ξ_g defined by $\xi_g(\omega) = g\xi(\omega)$ is a random G-variable and we have $\mathbb{E}(\xi_g) = g\mathbb{E}(\xi)$.

2.2 The Strong Law of Large Numbers (SLLN)

Definition 2.2. *Let $\xi_1, \ldots, \xi_n$ be a sequence of i.i.d. random G-variables with $\xi_i : \Omega \to G$ defined on a probability space $(\Omega, \mathcal{F}, P)$.*

- *The relative frequency*

$$\mu_n(g) = \mu_n(g, \omega) = \frac{|\{i \mid \xi_i(\omega) = g, 1 \leq i \leq n\}|}{n}$$

  is the probability with which g occurs in the random sample $\xi_1, \ldots, \xi_n$. μ_n defines a probability distribution on G.

- *The sampling weight function is the function $M_n : G \to \mathbb{R}$ defined by*

$$M_n(g) = \sum_{s \in G} d^2(g, s)\mu_n(s),$$

  where $d(g, s)$ is the distance between g and s in the Cayley graph $C_G(X)$ of G.

- *The sample mean-set of $\xi_1, \ldots, \xi_n$ is the set $\mathbb{S}_n$ defined by*

$$\mathbb{S}_n = \mathbb{S}(\xi_1, \ldots, \xi_n) = \{g \in G : M_n(g) \leq M_n(u), \forall u \in G\}.$$

We now state the SLLN generalized to graphs and groups which shows the convergence of the sample mean-set $\mathbb{S}_n$ to the mean-set $\mathbb{E}(\xi)$ when $n \to \infty$.

Theorem 2.1. *Let $G = \langle X \rangle$ be the group generated by a non empty set X, where its associated Cayley graph $C_G(X)$ is connected and locally finite. Let $\{\xi_i\}_{i=1}^{\infty}$ be a sequence of i.i.d. random G-variables. If the*

weight function $M_{\xi_1}(\cdot)$ *is totally defined and* $\mathbb{E}(\xi_1) = \{g\}$ *for some vertex* $g \in G$*, then*

$$\lim_{n\to\infty} \mathbb{S}(\xi_1, \ldots, \xi_n) = \mathbb{E}(\xi_1)$$

with probability 1.

For more details, see [14, 15].

Let $G = \langle X \rangle$ be a group and $G_1 = \{g_1, \ldots, g_n\}$ be a subset of group G with cardinality n. In [16], the following polynomial algorithm to compute the mean-set of G_1 is described.

Algorithm 2.1. ***Computation of the mean-set in a group***
INPUT: *the group* G *by its set* X *of generators and a subset* $G_1 = \{g_1, ..., g_n\}$ *of* G.
OUTPUT: *An element* g *of* G *having the smallest weight function.*
COMPUTATIONS:

A. *Choose a random element* $g \in G$ *according to some probability measure* μ *on* G.

B. *If for every* $x \in X^{\pm 1}, M_n(g) \leq M_n(gx)$*, then output* g.

C. *Otherwise put* $g \leftarrow gx$*, where* $x \in X^{\pm 1}$ *is an element minimizing the value of* $M_n(gx)$ *and go to step B.*

The computation of $\mathbb{S}_n$, the mean-set of the sample G_1, poses some problems:

- The computation of the set $\{M(g) : g \in G\}$ requires at least $O(|G_1|^2)$ elementary operations. This computation is practically impossible when n is too large;
- The computation of the distance function $d(\cdot, \cdot)$ is difficult in some groups like the braid groups.

However, Algorithm 2.1 presented above allows to solve the first problem. Indeed, it is a direct descent heuristic algorithm and it computes the sample mean-set since the weight function M_n comes from a sequence of random elements of G_1. The second problem is the computation of the distance between two elements in G. An approximation of the computation of the distance is described in [12]. Although it does not guarantee an optimal solution, this approximation sometimes has been used in a series of attacks.

3 Cryptanalysis of the authentication protocol

We now present the probabilistic approach used by Mosina to attack an authentication protocol in the braid groups [16].

3.1 The Sibert and *al.* authentication protocol

The authentication is a procedure that permits the user to convince the interlocutor of its identity. So, it involves two parties: the Prover (user) and the Verifier (interlocutor). The Prover provides the purported identity to the Verifier, and then both the Prover and the Verifier should corroborate and act simultaneously such that the Verifier should be convinced of the identity of the Prover. Only the Prover knows the secret value corresponding to his public one, and it is the proper use of this secret value which allows to convince the Verifier of its identity.

For $n \geq 2$, the n-string braid group denoted B_n is the group with the following presentation:

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{c} \sigma_i\sigma_j = \sigma_j\sigma_i \text{ for } |i-j| \geq 2 \\ \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j \text{ for } |i-j| = 1 \end{array} \right\rangle. \tag{1}$$

We present the Sibert and *al.* authentication protocol. The security level of this protocol is parametered by the size of the used braids and by the rank of the group B_n.

Protocol 3.1. *Let n be an integer, let b be a braid in B_n and let h be a hash function. b is written on its normal form or handle reduction form.*

Phase I. Keys generation
Private key: *Alice chooses a secret braid* $s \in B_n$
Public key: *Alice publishes* $(b, b^{'})$ *with* $b' = h(s^{-1}bs)$
Phase II. Authentication phase*: repeat* k *times*
Engagement: *Alice choses a random braid* r *and sends* $x = h(r^{-1}b'r)$ *to Bob;*
Challenge: *Bob sends a random bit* ϵ *to Alice;*
Answer:

- *If* $\epsilon = 0$*, Alice sends* $y = r$ *to Bob and Bob checks if* $x = h(y^{-1}b'y)$;
- *If* $\epsilon = 1$*, Alice sends* $y = h(sr)$ *to Bob and Bob checks if* $x = h(y^{-1}by)$.

3.2 Mean-set attack

In this section we present the mean-set attack on the protocol 3.1 described above (see also [16] or [14]).

3.2.1 Principle

If observe the Sibert and *al.* protocol 3.1, wee see that the Prover sends to the Verifier sequence of two types of random elements: r and sr, where r is a randomly generated element and s is the secret of the Prover. An Intruder (Eve) can intercept and arrange the answers of the challenges in a table similar to Table 1.

We obtain two sets R_0 and R_1 of elements, corresponding to $\epsilon = 0$ and $\epsilon = 1$ respectively. $R_0 = \{r_{i_1}, \ldots, r_{i_l}\}$ and $R_1 = \{sr_{j_1}, \ldots, sr_{j_t}\}$, where all the elements r_i $(i = 1, \ldots, k = l+t)$ are distributed according to a probability law μ. The objective of Eve is to retrieve the secret s using the intercepted sequences R_0 and R_1.

Suppose $G = \mathbb{Z}$. In this case, we write $R_1 = \{s+r_{j_1}, \ldots, s+r_{j_t}\}$, and we can compute the empirical average $\overline{r_0} = \frac{1}{l}\sum_{m=1}^{l} r_{i_m}$ of elements of $R_0 \subset \mathbb{Z}$ and the empirical average $\overline{r_1} = \frac{1}{t}\sum_{p=1}^{t}(s+r_{j_p}) = s+\frac{1}{t}\sum_{p=1}^{t} r_{j_p}$ of elements of $R_1 \subset \mathbb{Z}$. By the SLLN (Section 2.2, Theorem 2.1), if the

Table 1. Principle of the mean-set attack

Tour	Challenge	Answers type # 1	Answers type # 2
1	$\epsilon = 1$	–	sr_1
2	$\epsilon = 0$	r_2	–
3	$\epsilon = 0$	r_3	–
4	$\epsilon = 1$	–	sr_4
5	$\epsilon = 0$	r_5	–
...	...	...	...
k	$\epsilon = 0$	r_l	–

sequence R_0 is too large, then $\overline{r_0}$ tends to the mathematical expectation $\mathbb{E}(\mu)$ of the distribution μ in $\mathbb{Z}$. Similarly, if the sequence R_1 is too large, then $\overline{r_1}$ tends to $s + \mathbb{E}(\mu)$. Hence, by subtracting the limit of $\overline{r_0}$ to the limit of $\overline{r_1}$ we obtain an approximation of the secret s.

So, in this case, where $G = \mathbb{Z}$, we can compute the secret thanks to the following three properties:

(AV1) (SLLN for real-valued random variables): If $\{\xi_i\}_{i=1}^{\infty}$ is a sequence of real i.i.d. random variables and if $\mathbb{E}(\xi_1) < \infty$, then

$$\frac{1}{n}\sum_{i=1}^{n} \xi_i \to \mathbb{E}(\xi_1)$$

with probability 1 when $n \to \infty$.

(AV2) (Shift Property): For all real random variable ξ, we have

$$\mathbb{E}(c + \xi) = c + \mathbb{E}(\xi),$$

where c is a constant.

(AV3) (Efficient computation): The average $\frac{1}{n}\sum_{i=1}^{n} \xi_i$ is efficiently computable.

Now, this method can be generalized to some infinite groups where these three properties (AV1), (AV2) and (AV3) are defined similarly and are satisfied. Indeed let G be an infinite group.

- For a random G-variable $\xi : \Omega \to G$, define a set $\mathbb{E}(\xi) \subseteq G$ called *mean-set*;

- For a set of n random G-variables $\xi_1, \ldots, \xi_n$, define a set $\mathbb{S}_n = \mathbb{S}(\xi_1, \ldots, \xi_n) \subseteq G$ called the *sample mean-set* of $\xi_1, \ldots, \xi_n$.

Hence, we have the shift property $\mathbb{E}(s\xi) = s\mathbb{E}(\xi)$ and a generalization of the SLLN for groups in the sense that $\mathbb{S}(\xi_1, \ldots, \xi_n)$ converges to $\mathbb{E}(\xi_1)$ when $n \to \infty$, with probability 1. Moreover, suppose that the sample mean-set $\mathbb{S}(\xi_1, \ldots, \xi_n)$ is efficiently computable. Then Eve can form the sets $\mathbb{S}(sr_{j_1}, \ldots, sr_{j_{n-k}})$ and $\mathbb{S}(r_{i_1}, \ldots, r_{i_k})$ and compute

$$\mathbb{S}(sr_{j_1}, \ldots, sr_{j_{n-k}}) \cdot [\mathbb{S}(r_{i_1}, \ldots, r_{i_k})]^{-1},$$

which contains s with high probability when n is sufficiently large.

Below is the algorithm of the mean-set attack designed by Mosina.

3.2.2 Attack algorithm

Algorithm 3.1. The mean-set attack Algorithm
INPUT: *the Prover public key* (t, w) *and the sequences* R_0 *and* R_1;
OUTPUT: *an element* z *such that* $t = zwz^{-1}$ *or 'Failure'.*
COMPUTATION:

A. *Apply Algorithm 2.1 to* R_0 *and get* g_0.

B. *Apply Algorithm 2.1 to* R_1 *and get* g_1.

C. *If* $g_1 g_0^{-1}$ *satisfies* $t = (g_1 g_0^{-1})^{-1} w (g_1 g_0^{-1})$, *then retrieve* $g_1 g_0^{-1}$. *Otherwise output Failure.*

4 An amelioration of the mean-set attack

As mentioned by Sibert and *al.* in [21], we now consider the uniform law as the probability distribution used to generate r and ϵ in the protocol 3.1. We need to redefine the parameters of Section 2.2 for this restriction.

- Taking a sample S of k elements in B_n, the probability for an element to appear more than once in S is negligeable (since the probability distribution is uniform and $|B_n| = \infty$); we then have the *relative frequency*

$$\mu_k(g) = \mu_k(g, \omega) = \frac{1}{k},$$

  where $g \in S$.

- *The sample weight* is

$$M_k(g) = \frac{1}{k} \sum_{i \in S} d^2(g, i),$$

  where $d(\cdot, \cdot)$ is the distance function in B_n.

- *The sample mean-set* is

$$\mathbb{S}_k = \mathbb{S}(\xi_1, \ldots, \xi_k) =$$

$$= \{g \in B_n : \sum_{i \in S} d^2(g, i) \leq \sum_{i \in S} d^2(u, i), \quad \forall u \in B_n\}.$$

The SLLN is then stated as follows:

Theorem 4.1. *Let B_n be the n-string braid group and let $\{\xi_i\}_{i=1}^{\infty}$ be a sequence of i.i.d. random B_n-variables. If $M_{\xi_1}(\cdot)$ is totally defined and $\mathbb{E}(\xi_1) = \{g\}$ for an element $g \in B_n$, then*

$$\lim_{k \to \infty} \mathbb{S}_k = \mathbb{E}(\xi_1) = \{g\}.$$

Proof. Similar to the proof of theorem 2.1 which can be seen in [14, 15]. □

Now take a sequence of sample elements in B_n. We can approximate the mean-set of random B_n-variables. We experiment Mosina's algorithm 3.1 on CRAG by varying (n) the number of strings, (L) the length of the secret keys and (k) the number of elements in the sample. The results are presented in Tables 2–3.

T.T represents the ratio for obtaining the trivial braid e as element of the mean-set (on 100 tests).

$$T.T = \frac{|\{g_i | short(g_i) = e\}|}{100}$$

with g_i the element of the mean-set for the i-th test and $short(g_i)$ is the shortest normal or reduced element representing g_i.

DLMoy represents the average length of the braids when the element of the mean-set is different from the trivial braid.

$$DLmoy = \frac{1}{100 - TT * 100} \sum_{g \in S} l_X(g),$$

where S is the set of the elements which are different from the trivial braid and $l_X(g)$ represents the length of the element g with respect to X.

Table 2. Experimental results of the approximation of the mean-set of a random B_5-variable

L\k	20		40		80		160	
	T.T	**DLMoy**	**T.T**	**DLMoy**	**T.T**	**DLMoy**	**T.T**	**DLMoy**
10	65%	1,08	92%	1	100%	0	100%	0
20	45%	1,66	88%	1,8	96%	1	100%	0
30	45%	3,3	60%	1,7	89%	1,45	98%	1
40	21%	5,87	48%	5,11	64%	3,5	89%	8
50	14%	13,03	29%	6,7	71%	6,06	88%	5

An analysis of these results shows that the element of the mean-set of a random B_n-variable is either the trivial braid or either a braid which is very closed to the trivial braid (see that $DLMoy$ tends to 0

Table 3. Experimental results of the approximation of the mean-set of a random B_{10}-variable

L\k	20		40		80		160	
	T.T	**DLMoy**	**T.T**	**DLMoy**	**T.T**	**DLMoy**	**T.T**	**DLMoy**
10	85%	1,33	97%	1	100%	0	100%	0
20	49%	1,29	92%	0,99	100%	0	100%	0
30	46%	1,59	93%	1,01	100%	0	100%	0
40	31%	1,42	88%	1,66	97%	1	100%	0
50	29%	2,8	74%	1,8	98%	1	100%	0

when $T.T$ tends to 100). Hence we can deduce the following proposition:

Proposition 4.1. *Let B_n be the n-string braid group and let $g \in B_n$. Let $(\Omega, \mathcal{F}, P)$ be a probability space and let $\{\xi_i^g\}_{i=1}^{\infty}$ be a sample of random B_n-variables. Then for the random B_n-variable ξ_i^g defined by $\xi_i^g(\omega) = g\xi_i(\omega)$, we have*

$$\lim_{k\to\infty} \mathbb{S}_k(\xi_1^g, ...\xi_k^g) = g \lim_{k\to\infty} \mathbb{S}_k(\xi_1, ...\xi_k) = g.$$

This proposition means that $\lim_{n\to\infty} \mathbb{S}(\xi_1, \ldots, \xi_n) = \mathbb{E}(\xi_1) = e$, where e is the trivial braid in B_n. We then pose the following conjecture:

Conjecture 4.1. *Let B_n be the n-string braid group. Let $(\Omega, \mathcal{F}, P)$ be a probability space and let $\xi : \Omega \to B_n$ a random B_n-variable. Then $\mathbb{E}(\xi) = \{g\}$, where the normal form $short(g)$ of g is such that $short(g) = e$, the trivial braid in B_n.*

Thus, from the set R_1 defined in Section 3.2.1, one can compute the set

$$\mathbb{S}(sr_{j_1}, \ldots, sr_{j_k})$$

which contains element s with a very high probability when k is large.

We can then rewrite the attack Algorithm 3.1 as follows.

Algorithm 4.1. The revisited mean-set attack Algorithm
INPUT: *the Prover public key* (t, w) *and a sequence* R_1
OUTPUT: *an element* g *such that* $t = gwg^{-1}$ *or 'Failure'*
COMPUTATION:

A. Apply Algorithm 2.1 to R_1 *and get* g.

B. If g *satisfies* $t = g^{-1}wg$, *then retrieve* g. *Else Failure.*

The experimental results, implemented on CRAG with this revisited mean-set attack Algorithm 4.1, are presented in Tables 4–6. Here, we vary n the number of strings, the length L of the words and the number k of tours in the algorithm (or elements in the sample).

Table 4. Experimental results of the attack Algorithm 4.1 in B_5

L\k	**20**	**40**	**80**	**160**
10	66%	95%	100%	100%
20	55%,	85%,	95%	100%
30	13%	38%	67%,	100%

Table 5. Experimental results of the attack Algorithm 4.1 in B_{10}

L\k	**20**	**40**	**80**	**160**
10	73%	99%	100%	100%
20	60%,	95%,	100%	100%
30	45%	90%	100%,	100%

Table 6. Experimental results of the attack Algorithm 4.1 in B_{20}

L\k	**20**	**40**	**80**	**160**
10	94%	100%	100%	100%
20	89%,	100%,	100%	100%
30	65%	97%	100%,	100%

On these tables, we see that the rate of success increases when the values of k increase. The length of the key influences the rate of success. Also, the success rate increases with the rank (number of

strings) of the group . These results, as those on Tables 2 and 3, confirm Mosina and Ushakov's obtained in [16]. Moreover we obtain a slight rise of the success rate, compared to Mosina. Furthermore, we gain in computation time since we need to compute only the mean-set of the set R_1, instead of computing for R_0 and R_1. Note that the computation of the mean-set is timely significant when k and L are large.

Acknowledgement

This work was completed at Saint Francis University (Loretto, PA, USA) during the second author's visit thanks to the Fulbright Program. Saint Francis University and the Fulbright Scholar Program are gratefully acknowledged.

References

[1] CRyptography And Groups (CRAG) C++ Library, available at http://www.acc.stevens.edu/downloads.php.

[2] P. Dehornoy. *Efficient solutions to the braid isotopy problem*, Disc. Appl. Math., Volume 156 Issue 16, September, (2008) 3091–3112, (online: http://www.arxiv.org/abs/math.GR/0703666).

[3] P. Dehornoy. *Using shifted conjugacy in braid-based cryptography*, Contemp. Math. 418 (2006) 65–73.

[4] W. Diffie, M.E. Hellman. *New directions of cryptography*, IEEE Transactions on Information theory, **22** (1976) 644–654.

[5] A. Groch, D. Hofheinz, R. Steinwandt. *A practical attack on the root problem in braid groups*, Contemp. Math. **418** (2006) 121–131.

[6] D. Hofheinz, R. Steinwandt. *A practical attack on some braid group based cryptographic primitives*, PKC 2003; Springer Lect. Notes in Comp. Sci. **2567** (2002) 187–198.

[7] C. Kassel, V. Turaev. *Braid groups*, Springer, 2007.

[8] S. Lal, A. Chaturvedi. *Authentication schemes using braid groups*, preprint (2005) (online: http://arxiv.org/pdf/cs.CR/0507066).

[9] J. Longrigg, A. Ushakov. *A Practical Attack on a Certain Braid Group Based Shifted Conjugacy Authentication Protocol*, Groups-Complexity-Cryptology, Vol. 1, No. 2 (2009) 275-286.

[10] S. Maffre. *Reduction of conjugacy problem in braid groups, using two Garside structures*, WCC 2005, 214–224.

[11] A. G. Miasnikov, V. Shpilrain, A. Ushakov *A practical attack on some braid group based cryptographic protocols.* Advances in Cryptology – CRYPTO 2005, Lecture Notes in Computer Science 3621, pp. 86–96. Springer, Berlin, 2005.

[12] A. G. Miasnikov, V. Shpilrain, A. Ushakov. *Random Subgroups of Braid Groups: An Approach to Cryptanalysis of a Braid Group Based Cryptographic Protocol.* Advances in Cryptology – PKC 2006, Lecture Notes in Computer Science 3958, pp. 302–314. Springer, Berlin, 2006.

[13] A. G. Miasnikov, V. Shpilrain, A. Ushakov. *Group-based Cryptography*, Advanced Courses in Mathematics - CRM Barcelona. Birkhäuser Basel, 2008.

[14] N. Mosina. *Probability on graphs and groups: theory and applications*, Ph.D. thesis, Columbia University, 2009. Available at http://www.math.columbia.edu/ thaddeus/theses/2009/mosina.pdf.

[15] N. Mosina, A. Ushakov. *Strong law of large numbers on graphs and groups*, Groups Complexity Cryptology, Vol. 3 Issue 1 (2011) 67–103.

[16] N. Mosina, A. Ushakov. *Mean-set attack: cryptanalysis of Sibert and al. authentication protocol*, J. Math. Cryp. 4, (2010) 149–174.

[17] R.L Rivest, A. Shamir, L.M Adleman. *A method for obtaining digital signatures and public-key cryptosystems*, Communication of ACM, **21** (1978) 120–126.

[18] C.E. Shannon. *Communication theory of secrecy systems*, Bell System Technical Journal, **28** (1949) 656–715.

[19] P.W. Shor. *Polynomial-time algorithm for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comp. **26**(5) (1997) 1484–1509.

[20] V. Shpilrain, A. Ushakov. *An authentication scheme based on the twisted conjugacy problem*, Applied Cryptography and Network Security, Lecture Notes in Computer Science Volume 5037, (2008) 366–372.

[21] H. Sibert, P. Dehornoy, M. Girault. *Entity authentication schemes using braid word reduction*, Proc. Internat. Workshop on Coding and Cryptography, 153–164, Versailles, 2003 OR Discrete applied Mathematics 154, (2006), 420–436.

[22] M.R. Spiegel. *Probabilits et statistique*. Neuvime tirage, MC Graw-Hill, Paris (1992).

[23] B. Tsaban. *On an authentication scheme based on the root problem in the braid group*, arXiv:cs/0509059 v2, (2007)

[24] M. Welschenbach. *Cryptography in C and C++* , Second Edition, 2005.

Sidoine Djimnaibeye, Daniel Tieudjo,
Norbert Youmbi

Received May 20, 2015

Sidoine Djimnaibeye, Daniel Tieudjo
Department of Mathematics and computer science
The University of Ngaoundere
P.O. Box 454 Ngaoundere – Cameroon
E–mail: dosiusher@yahoo.fr, tieudjo@yahoo.com

Norbert Youmbi
School of Science, Department of Mathematics
Saint Francis University
117 Evergreen Dr, Loretto PA, 15940 USA
E–mail: NYoumbi@francis.edu

Computer Science Journal of Moldova, vol.**23**, no.3(69), 2015

Classification of Early Stages of NAFLD Based on Dual Diagnostic Methods*

Iulian Secrieru, Svetlana Cojocaru, Constantin Gaindric, Olga Popcova, Svetlana Ţurcan

Abstract

High prevalence of non-alcoholic fatty liver disease (NAFLD) has made this domain of medical diagnostics one of high professional and public interest. The major problem of NAFLD diagnostics is that in its initial phase non-alcoholic fatty liver tends to be benign without tendency to progress, while in its second phase – non-alcoholic steatohepatitis (NASH) can progress to cirrhosis, which subsequently may cause hepatocellular carcinoma. This fact explains the need for more sensitive classifications that would allow early diagnostics of NAFLD. NAFLD diagnostics in most cases is based on clinicopathological criteria – decision rules expressed through ultarasound signs and laboratory data, annotated by hepatologist/gastroenterologist. In this article we describe the process of creation of a classification of NAFLD early stages based on a decisional reasoning, which combines two methods of medical diagnostics.

Keywords: Non-alcoholic fatty liver, NAFLD diagnostics, medical ultrasound, hepatologist/gastroenterologist, pathological liver states classification.

*This research is supported by ASM-BMBF project Nr. 13.820.18.02/GA "Computer-Aided Tools for Diagnostics and Classification of Early Stages of Non-Alcoholic Fatty Liver Disease with Predictive Models for Risk Assessment of Complications (CATDC-NAFLD)".

1 Introduction

NAFLD in its initial stage is asymptomatic, often with normal hepatic functional tests, and is difficult to diagnose [1]. It is estimated that NAFLD is the most common liver disease both in countries of Western Europe and United States with high income per capita and in poor countries [2].

NAFLD encompasses a broad spectrum of liver diseases. NAFLD is a disease that can progress, and in its natural evolution passes through several stages. It begins with free fatty acids synthesis in the liver, resulted in steatosis. Simple steatosis refers to diseases with a favorable course and possibility of a complete regression. However, in 10-20% of cases steatosis is associated with inflammation, which leads to non-alcoholic steatohepatitis [1]. In present, hepatic steatosis and steatohepatitis are considered early stages of NAFLD. The evolution of fibrosis, in turn, causes transformation into cirrhosis and cancer – advanced stages of NAFLD [2].

The degree of filling of the liver with fat can be determined using sonography technique and/or laboratory tests.

Both sonographists and hepatologists/gastroenterologists examine separately at best only functionality of the liver (most often the whole hepato-pancreato-biliary region), rather than the degree of steatosis. The degree of steatosis appears in the patients' diagnosis only if the physician finds NASH fibrosis (at best of degree 1). This is also explained by the fact that generally a hepatologist examines patients with suspected NAFLD only if they are referred by a family doctor or sonographist.

Therefore, it is impossible to determine exactly interconnection between the steatosis degree and fibrosis appearance, and the role of steatosis as a trigger.

Additionally, at the moment, there is no specialized examination algorithm and protocol (within the meaning of physicians) for patients with suspected NAFLD or patients with early stages of NAFLD.

In this paper we propose a classification using two types of findings (sonographic and hepatologic) to estimate the steatosis degree at early stages.

2 Related work

In a general way NAFLD diagnostics can be reduced to: (i) excluding the fact of alcohol consumption or its daily limitation by less than 20 g for women and less than 30 g for men; (ii) determining the presence of hepatic steatosis by imaging technigues or laboratory tests; and (iii) excluding of other liver diseases.

There are several methods of medical imagistics that can identify the degree of infiltration of the liver with fat. Ultrasound is reliable and accurate method for diagnostics of the diseases of hepato-pancreato-biliary region with sensitivity more than 70-80%. Another important advantage is that ultrasound diagnostics is by far inexpensive compared to many other imaging methods of diagnostics like MRI, CT, etc.

However, this diagnostics technique has its own drawbacks. The accuracy of ultrasound in detecting pathologies is a good one, but has some limitations, because of both false-positive and false-negative results. In addition, ultrasound images are noisy, blurred in shape, and suffer from echoes.

So, the first problem is to obtain a good image, the most relevant and useful for physician's decision-making. This is the main task of an operator, and the reason for which ultrasound investigation is considered highly operator dependent.

Ultrasound diagnostics of pathological modifications is based on the analysis of characteristic signs from images, obtained for the investigated organ. The resulting conclusion is quite subjective, and widely depends on a physician's experience. So, the second, and probably more important problem, is the interpretation of the obtained ultrasound images.

In order to solve two major problems described above, SonaRes methodology and technology for formalization of professional expert knowledge in the domain of medical ultrasound disgnoctics of hepato-pancreatic-biliary region were proposed in [3].

The SonaRes knowledge base includes the following data and expert knowledge:

- the knowledge base for gallbladder contains 335 facts and 54 de-

cision rules, 166 model images annotated by the expert group, 226 images with regions of interest (ROIs) marked;

- for pancreas – 231 facts, 52 decision rules, 106 model images, 137 images with ROIs marked;
- for liver – 167 facts, 31 decision rules, 87 model images, 111 images with ROIs marked;
- for bile ducts – 257 facts, 15 decision rules, 30 model images, 37 images with ROIs marked.

On the other hand, hematologists/gastroenterologists in their professional practice use various scoring systems, which describe the hepatic functionality.

The authors of the NAFLD activity score (NAS) – Matteoni, Brunt, and the NASH Clinical Research Network Pathology Committee – proposed the best-known pathological classification of NAFLD/NASH [4-5].

The NAS represents an unweighted sum of the scores for steatosis (0-3), lobular inflammation (0-3) and ballooning degeneration (0-2). Scores of 5 or more are correlated well with the diagnosis of NASH, as confirmed by an experienced pathologist who studied the specimens independently. Scores of less than 3 are correlated equally well with "not NASH", while scores of 3 or 4 did not allow clear assignments to one or the other category.

NAS differentiates fibrosis in four stages: stage 1 means perisinusoidal fibrosis in zone 3; stage 2 is characterized by perisinusoidal and portal/periportal fibrosis; stage 3 is defined as bridging fibrosis; stage 4 reflects cirrhosis.

3 Classification of early stages of NAFLD in CATDC-NAFLD

Scenarios, describing NAFLD progress from hepatic steatosis (grade 1, grade 2, grade 3), to non-alcoholic steatohepatitis (fibrosis 0-1, fibrosis

2-3) and cirrhosis are known, and are presented in Fig. 1. Fibrosis progression in stages 0-1-2-3 is reversible, while there is no reversibility of fibrosis stage 4 (cirrhosis).

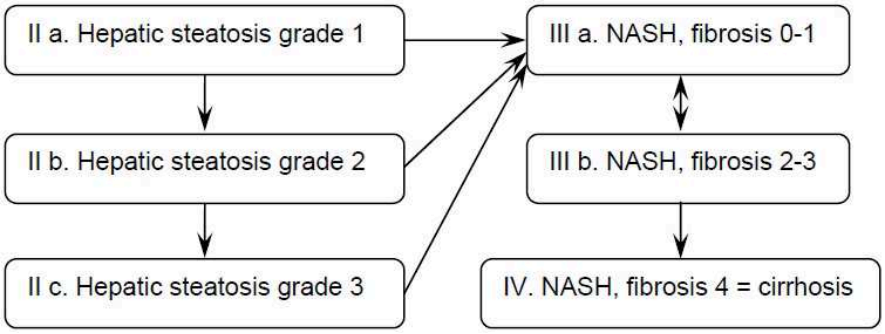

Figure 1. Scenarios, describing NAFLD progress

But, at present there is neither generally accepted theory on NAFLD pathogenesis, nor complete understanding of mechanisms of NAFLD onset and progress (its transition from steatosis to steatohepatitis). Therefore, the knowledge describing NAFLD diagnostics domain is needed to be formalized.

Discovering and formalization of knowledge diagnostics process, onset and progress of early stages of NAFLD is one of goals of CATDC-NAFLD project [6] – development of computer-aided tools for diagnostics and classification of early stages of NAFLD.

To create the knowledge base of CATDC-NAFLD we have used SonaRes technology [3], which allows to incorporate the kernel of the SonaRes knowledge base about liver pathologies into CATDC-NAFLD. This kernel includes the following data and expert knowledge: 207 facts, 38 decision rules, 81 model images, 111 images with ROIs marked.

After that, sonographic and hepatologic experts extended the number of liver rules from 38 into 44, taking into account NAFLD specifics. For diffuse hepatic steatosis 3 rules were created, identifying mild, moderate and severe forms. Also 3 rules were created for steatohepatitis, separating liver fibrosis into stages F0-F1, F2-F3 and F4. As the result

11 rules (including liver normal state) and 120 facts, corresponding to NAFLD pathologies, were selected from 44 rules and 207 facts (see Table 1).

Table 1. CATDC-NAFLD knowledge base evolution (sonographic findings)

	SonaRes liver KB	CATDC-NAFLD KB adjusted to NAFLD diagnostics
Total facts	207	120
Total rules	38	11
Total model US images	81	24
Total US images with ROIs marked	111	31

The obtained 11 rules are based on sonographic findings and formulated using sonographic terminology. It can lead to misunderstanding and wrong interpretation by hepatologist.

In order to solve this problem we, together with the experts-physicians, established a correspondence between liver sonographic conclusions (rules) and NAFLD pathological states (IIa-IIc, IIIa-IIIb, IV) (see Table 2).

Validation of the obtained correspondence between liver sonographic conclusions and NAFLD pathological states was done on 10 patients.

As a result of the validation process the need for the addition of hepatologic findings based on laboratory tests was identified. It will help to make the desired classification more accurate.

An analysis of 53 liver protocols was done in order to determine characteristics, which are specific to NAFLD. The following four groups of hepatologic characteristics were identified: general data, risk factors, clinical data, and laboratory tests. The total number of hepatologic findings was essensially redused for the case of NAFLD (see Table 3).

Thus, the base of professional knowledge, used in the diagnostics

Table 2. Classification of NAFLD pathological states

	Liver sonographic conclusion		NAFLD pathological state
1	R00. Normal liver	I	Predisposition to NAFLD
2	R20.1. Metabolic diseases. Diffuse hepatic steatosis, mild form	II a	Hepatic steatosis grade 1
3	R20.2. Metabolic diseases. Diffuse hepatic steatosis, moderate form	II b	Hepatic steatosis grade 2
4	R20.3. Metabolic diseases. Diffuse hepatic steatosis, severe form	II c	Hepatic steatosis grade 3
5	R20a. Metabolic diseases. Parcelar hepatic steatosis	II	Hepatic steatosis
6	R20b. Metabolic diseases. Diffuse hepatic steatosis with sparing areas. Focal fatty sparing	II	Hepatic steatosis
7	R20c. Metabolic diseases. Focal hepatic steatosis (pseudotumoral)	II	Hepatic steatosis
8	R24a. Steatohepatitis. Liver fibrosis F0-F1	III a	NASH, fibrosis 0-1
9	R24b. Steatohepatitis. Liver fibrosis F2-F3	III b	NASH, fibrosis 2-3
10	R24c. Steatohepatitis. Liver fibrosis F4. Liver cirrhosis	IV	NASH, fibrosis 4 = cirrhosis
11	R35. Portal hypertension	III b IV	NASH, fibrosis 2-3, NASH, fibrosis 4

of early stages of NAFLD by both sonographists and hepatologists, was obtained. This knowledge base consists of 120 facts describing NAFLD, 18 facts describing NAFLD onset risk factors and 11 decision rules. Using this knowledge base and inference engine of the SonaRes technology, early stages of NAFLD can be classified.

Table 3. CATDC-NAFLD knowledge base evolution (hepatologic findings)

	Total hepatologic findings	NAFLD-related findings
General data	6	4
Risk factors (no alcohol-related, used only for NAFLD progression scenarios)	29	18
Clinical data	36	2
Laboratory tests	55	14
Total	126	20 (+18)

4 Conclusions and future work

The current practice is the following: a hepatologist generally only assesses patients after they are referred by the ultrasound physician. But not always ultrasound images can show the liver pathological changes in the early stages of NAFLD, or ultrasound physician pays attention to more serious pathology in gallbladder, pancreas and biliary system. It is a major cause of delayed access to the hepatologist and late NAFLD diagnostics. The proposed classification allows to focus attention on the early liver pathological changes, revealed not by one, but by two main diagnostics methods, that improves the accuracy of diagnosis and allows to diagnose early stages of NAFLD. In addition, this classification forces ultrasound physician and hepatologist to cooperate at NAFLD early stages manifestation.

As in the inference mechanism we use decision rules that involve sonographic findings, obtained from phisicians-experts, and case-based hepatologic findings, it is necessary to validate the obtained inference on the same cohort. Another task for the future work is to create a score, taking into account both sonographic and hepatologic findings.

References

[1] P. Loria, et al. *Practice guidelines for the diagnosis and management of nonalcoholic fatty liver disease.* A decalogue from the Italian Association for the Study of the Liver (AISF) Expert Committee. Dig Liver Dis. 42(4) (2010), pp. 272–82.

[2] J.R. Lewis, et al. *Non-alcoholic Fatty Liver Disease: A Review and Update.* Digestive Diseases and Sciences, 55 (2010), pp. 560–578.

[3] L. Burtseva, S. Cojocaru, C. Gaindric, O. Popcova, Iu. Secrieru. *SonaRes – Diagnostic Decision Support System for Ultrasound Examination.* Proceedings of the 3rd International Conference on e-Health and Bioengineering EHB 2011, Iasi, Romania, November 24-26 (2011), pp. 239–242.

[4] E.M. Brunt. *Nonalcoholic steatohepatitis: definition and pathology.* Semin Liver Dis, 21 (2001), pp. 3–16.

[5] D.E. Kleiner, E.M. Brunt, M. Van Natta et al. *Nonalcoholic Steatohepatitis Clinical Research Network. Related Articles: design and validation of histological scoring system for nonalcoholic fatty liver disease.* Hepatology, 41 (2005), pp. 1313–21.

[6] C. Gaindric, Iu. Secrieru, O. Popcova, S. Ţurcan. *Concept of computer-aided tools for diagnostics and classification of early stages of non-alcoholic fatty liver disease.* Proceedings IIS 2013 – International Conference on Intelligent Information Systems, Chisinau, Moldova, August 20-23 (2013), pp. 94–99.

Iulian Secrieru, Svetlana Cojocaru,
Constantin Gaindric, Olga Popcova, Svetlana Ţurcan

Received November 2, 2015

Iulian Secrieru
Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova
5 Academiei Street, Chisinau, MD 2028
Phone: (373 22) 73-81-30
E–mail: secrieru@math.md

Svetlana Cojocaru
Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova
5 Academiei Street, Chisinau, MD 2028
Phone: (373 22) 72-84-14
E–mail: svetlana.cojocaru@math.md

Constantin Gaindric
Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova
5 Academiei Street, Chisinau, MD 2028
Phone: (373 22) 72-50-12
E–mail: gaindric@math.md

Olga Popcova
Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova
5 Academiei Street, Chisinau, MD 2028
Phone: (373 22) 73-81-30
E–mail: oleapopcova@yahoo.com

Svetlana Ţurcan
State University of Medicine and Pharmacy Nicolae Testemitanu
165 Stefan cel Mare Ave., Chisinau, MD 2004
E–mail: veisa@mail.ru

Annotation on PhD Thesis

Title: On the Power and Universality of Biologically-inspired Models of Computation

Author: Sergiu Ivanov

Institute: Université Paris Est, France

Defence date: June 23, 2015

Keywords: computational completeness, universality, insertion-deletion systems, networks of evolutionary processors, controlled insertion-deletion systems, register machines, Petri nets.

Objective: Consider the problems of computational completeness and universality for several biologically-inspired models of computation: insertion-deletion systems, networks of evolutionary processors, and multiset rewriting systems. The presented results fall into two major categories: study of expressive power of the operations of insertion

and deletion with and without control, and construction of universal multiset rewriting systems of low descriptional complexity.

Novelty: The thesis focuses on novel models of computation inspired by the biological cell and gives original techniques of analysis of the expressive power of these models as well as of optimisation of their descriptional complexity.

Results: In the first part of the thesis we focus on insertion-deletion systems and we show that allowing one-symbol insertion and deletion rules to check a two-symbol left context enables them to generate all regular languages. Moreover, we prove that allowing longer insertion and deletion contexts does not increase the computational power. We further consider insertion-deletion systems with additional control over rule applications and show that the computational completeness can be achieved by systems with very small rules.

The second part of the thesis is concerned with the universality problem, which consists in finding a fixed element able to simulate the work any other computing device. We start by considering networks of evolutionary processors (NEPs), a computational model inspired by the way genetic information is processed in the living cell, and construct universal NEPs with very few rules. We then focus on multiset rewriting systems, which model the chemical processes running in the biological cell. For historical reasons, we formulate our results in terms of Petri nets. We construct a series of universal Petri nets and give several techniques for reducing the numbers of places, transitions, inhibitor arcs, and the maximal transition degree. Some of these techniques rely on a generalisation of conventional register machines, proposed in this thesis, which allows multiple register checks and operations to be performed in a single state transition.

Software contributions: The construction of small universal Petri nets (multiset rewriting systems) was supported by software developed by the author with the goal of automating solutions to some combinatorial and optimisation problems. Programmatic definitions of

the discussed models of computing were given and are available online[1].

Applicative value: The applicative value of the thesis has two principal aspects. On the one hand, the results concerning insertion-deletion systems contribute directly to understanding of the complexity of the biological processes this model was inspired by, whereas the described small universal objects may serve as a theoretical foundation for the construction of simple but computationally universal biological computers. On the other hand, the thesis proposes a number of original approaches to the discussed problems, which can be generalised and applied in a different context.

[1] `https://github.com/scolobb/computing-devices`