Computer Science Journal of Moldova, vol.**25**, no.2(74), 2017

Nominative data with ordered set of names

Volodymyr G. Skobelev, Ievgen Ivanov, Mykola Nikitchenko

Abstract

In the paper we analyze the set of nominative sets, which can be considered as some mathematical model for data used in computing systems, under assumption that the set of names is linearly ordered. We design algorithms, implemented for execution of basic set-theoretic operations on this set of nominative sets under assumption that nominative sets are presented by doubly linked lists with the order of names in increasing strength. The worst-case time complexity under logarithmic weight for the designed algorithms is investigated in detail. Applications of presented results for table algebras, which are mathematical models, intended for developing and theoretic analysis of relational databases, as well as of associated query languages, are proposed. The obtained results can be used in formal software development.

Keywords: nominative set, nominative data, linear ordering, set-theoretic operations, time complexity of algorithms.

1 Introduction

It's well known that research of deep internal links "between the theory of programming and the products of software engineering practice" [1] is one of the main challenges of the 21^{st} century. The significance and complexity of this problem is caused by variety of theoretical and application-oriented approaches, that are hardly comparable with each other [2–11].

We deal with composition-nominative approach to program formalization [12–14]. Informally speaking, any software is regarded as some data processor. Mathematical model intended to present different data structures, used in computing systems (arrays, lists, tables, trees, etc.),

in a unified form is *a nominative set*. This model is based on the notion of *a name-value relation*. The cost of this universality is high complexity in elaboration of formal theory for nominative sets, intended to automate software design, starting with creation of formal specifications, and finishing with resolving problems of verification and testing. The main reason of this complexity is caused by the factor that it is necessary to deal with algebraic structures, which significantly differ from the classic ones.

This situation has been illustrated accurately in [15], where the most general case, arising under the assumption that the sets of names and data are abstract ones, has been investigated. Indeed, it has been established that for algebraic system in which the carrier is any fixed set of all nominative sets, and the set of operations is the set of set-theoretic operations over these nominative sets, there exist the following different types of subalgebras: commutative and non-commutative semigroups, non-commutative non-associative magma, and semi-rings. Also it has been established that any fixed partially ordered set of all nominative sets is the union of the set of overlapping isomorphic maximal closed intervals. The mappings that define isomorphism between two intervals differ significantly from each other, and the family of these mappings has sufficiently complicated structure.

The present paper is further development of investigations, that has been started in [15]. We study any fixed set of all nominative sets under assumption that the set of names is linearly ordered. The paper is organized as follows: in Section 2 we recall necessary notions and definitions; in Section 3 we investigate time complexity of set-theoretic operations on any fixed set of all nominative sets; in Section 4 we illustrate how results, established in Section 3 can be applied in table algebras; in Section 5 we give conclusions.

2 Basic notions

Let V ($|V| \geq 2$) and A ($|A| \geq 2$) be finite sets of names and data, respectively. The set $\mathfrak{F}_{V,A}$ of all (possibly, partial) mappings from V to A is called the set of nominative sets. Basic set-theoretic operations

on the set $\mathfrak{F}_{V,A}$ are defined as follows:

$$f_1 \cap f_2 = f \Leftrightarrow graph(f_1) \cap graph(f_2) = graph(f), \quad (1)$$

$$f_1 \setminus f_2 = f \Leftrightarrow graph(f_1) \setminus graph(f_2) = graph(f), \quad (2)$$

$$f_1 \cup f_2 = f \Leftrightarrow graph(f_1) \cup graph(f_2) = graph(f), \quad (3)$$

$$f_1 \oplus f_2 = f \Leftrightarrow graph(f_1) \oplus graph(f_2) = graph(f), \quad (4)$$

$$f_1 \triangleright f_2 = f \Leftrightarrow graph(f_1) \cup graph(f_2|_{\mathrm{Dom} f_2 \setminus \mathrm{Dom} f_1}) = graph(f), \quad (5)$$

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

$$\Leftrightarrow graph(f_1|_{\mathrm{Dom} f_1 \setminus \mathrm{Dom} f_2}) \cup graph(f_2|_{\mathrm{Dom} f_2 \setminus \mathrm{Dom} f_1}) = graph(f). \quad (6)$$

We recall that $\cup$ and $\oplus$ are partial operations on the set $\mathfrak{F}_{V,A}$.

It is supposed that some linearly ordering relation $<_V$ is fixed on the set of the names V (this is true, for example, when the set V reflects different types of memory addressing, such as real, physical, flat, or absolute addressing). Thus, we get the possibility to present any nominative set $f = \{(v_i, a_i) | i = 1, \ldots, k\} \in \mathfrak{F}_{V,A}$ $(k = 0, 1, \ldots, |V|)$ by such doubly linked list L_f:

$$\begin{array}{c} \\ i_1 \\ i_2 \\ \ldots \\ i_{k-1} \\ i_k \end{array} \begin{array}{c} \begin{array}{cccc} \text{previous} & \text{name} & \text{data} & \text{next} \end{array} \\ \left(\begin{array}{cccc} * & v_{r_1} & a_{r_1} & i_2 \\ i_1 & v_{r_2} & a_{r_2} & i_3 \\ \ldots & \ldots & \ldots & \ldots \\ i_{k-2} & v_{r_{k-1}} & a_{r_{k-1}} & i_k \\ i_{k-1} & v_{r_k} & a_{r_k} & * \end{array} \right), \end{array}$$

that the inequalities $v_{r_1} <_V v_{r_2} <_V \ldots <_V v_{r_{k-1}} <_V v_{r_k}$ hold.

In the next section we will show that for basic set-theoretic operations on the set $\mathfrak{F}_{V,A}$ the order of names in increasing strength also remains in the resulting doubly linked list.

For doubly linked list L_f we set:

$$\mathsf{L}_f(i_j) = \begin{cases} (*, v_{r_1}, a_{r_1}, i_2), & if \ \ j = 1 \\ (i_{j-1}, v_{r_j}, a_{r_j}, i_{j+1}), & if \ \ j = 2, \ldots, k-1 \\ (i_{k-1}, v_{r_k}, a_{r_k}, *), & if \ \ j = k \end{cases}$$

and $length(\mathsf{L}_f) = k$. The last equality implies that

$$length(\mathsf{L}_f) \leq |V| \tag{7}$$

for any nominative set $f \in \mathfrak{F}_{V,A}$.

3 Time complexity of set-theoretic operations on the set $\mathfrak{F}_{V,A}$

We will investigate time complexity of set-theoretic operations (1)-(6) on the set $\mathfrak{F}_{V,A}$ under the following assumptions:

1) initial nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ are presented by the doubly linked lists, respectively L_{f_1} and L_{f_2};

2) the result of operation $f_1 \diamond f_2$ ($\diamond \in \{\cap, \setminus, \cup, \oplus, \triangleright, \boxplus\}$) is the doubly linked list $\mathsf{L}_{f_1 \diamond f_2}$;

3) the address parameter is denoted:

– by $i_j^{(1)}$ ($j = 1, \ldots, length(\mathsf{L}_{f_1})$), for doubly linked list L_{f_1};

– by $i_j^{(2)}$ ($j = 1, \ldots, length(\mathsf{L}_{f_2})$), for doubly linked list L_{f_2};

– by $i_j^{(3)}$ ($j = 1, \ldots, length(\mathsf{L}_{f_1 \diamond f_2})$), for doubly linked list $\mathsf{L}_{f_1 \diamond f_2}$.

Analyzing time complexity of set-theoretic operations on the set $\mathfrak{F}_{V,A}$, we deal with asymptotic worst-case time complexity of algorithms under logarithmic weight [16]. The last factor implies that:

1) time $T = O(\log |V|)$ ($|V| \to \infty$) is needed to check for any names $v_1, v_2 \in V$, what of formulae, either $v_1 = v_2$, $v_1 \neq v_2$, or $v_1 <_V v_2$, holds;

2) time $T = O(\log |A|)$ ($|A| \to \infty$) is needed to check for any data $a_1, a_2 \in A$, what of formulae, either $a_1 = a_2$, or $a_1 \neq a_2$, holds;

3) time $T = O(\log length(\mathsf{L}_f))$ ($length(\mathsf{L}_f) \to \infty$) is needed for transition (if it is possible) from any current element of any doubly linked list L_f, either to its previous element, or to its next element.

If initial nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ are presented by doubly linked lists, then the idea of how to compute the result of operation $f_1 \diamond f_2$ ($\diamond \in \{\cap, \setminus, \cup, \oplus, \triangleright, \boxplus\}$), is rather simple: sequentially moving through the doubly linked lists L_{f_1} and L_{f_2}, from their beginnings to their ends, we create the doubly linked list $\mathsf{L}_{f_1 \diamond f_2}$. However, despite

transparency of this idea, there are many subtle aspects in case of its implementation. For this reason, we will design appropriate algorithms in an explicit form.

The operation $\cap$ on the set $\mathfrak{F}_{V,A}$ can be implemented as follows.

ALGORITHM 1.
Input: doubly linked lists L_{f_1} and L_{f_2}.
Output: doubly linked list $\mathsf{L}_{f_1 \cap f_2}$.
Step 1. $\mathsf{L}_{f_1 \cap f_2} := \emptyset$, $j_3 := 0$.
Step 2. If $length(\mathsf{L}_{f_1}) = 0$ or $length(\mathsf{L}_{f_2}) = 0$,
then HALT, else $j_1 := 1$, $j_2 := 1$.
Step 3. $\mathbf{u}_1 := \mathsf{L}_{f_1}(i^{(1)}_{j_1})$, $\mathbf{u}_2 := \mathsf{L}_{f_2}(i^{(2)}_{j_2})$.
Step 4. If $\mathrm{pr}_2\mathbf{u}_1 = \mathrm{pr}_2\mathbf{u}_2$, then go to step 7.
Step 5. If $\mathrm{pr}_2\mathbf{u}_1 <_V \mathrm{pr}_2\mathbf{u}_2$, then $r := 1$, else $r := 2$.
Step 6. Call *Procedure1.1*(r).
Step 7. If $\mathrm{pr}_3\mathbf{u}_1 = \mathrm{pr}_3\mathbf{u}_2$, then Call *Procedure1.2*
Step 8. If $\mathrm{pr}_4\mathbf{u}_1 = *$ or $\mathrm{pr}_4\mathbf{u}_2 = *$,
then HALT, else $j_1 := j_1 + 1$, $j_2 := j_2 + 1$,
and go to step 3.

Procedure1.1(r)
begin;
If $\mathrm{pr}_4\mathbf{u}_r = *$, then HALT,
else $j_r := j_r + 1$, $\mathbf{u}_r := \mathsf{L}_{f_r}(i^{(r)}_{j_r})$, and go to step 4;
end;

Procedure1.2
begin;
If $j_3 \neq 0$, then go to M1;
$j_3 := j_3 + 1$, $\mathsf{L}_{f_1 \cap f_2}(i^{(3)}_{j_3}) := (*, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$,
go to step 8;
M1: $\mathrm{pr}_4\mathsf{L}_{f_1 \cap f_2}(i^{(3)}_{j_3}) := j_3 + 1$,
$\mathsf{L}_{f_1 \cap f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$, $j_3 := j_3 + 1$,
go to step 8;
end;

Correctness of algorithm 1 is justified by the following two factors. Firstly, algorithm 1 always halts. Secondly, the list $\mathsf{L}_{f_1 \cap f_2}$ consists of those and only those pairs $(v, a) \in V \times A$, which are elements of the set $graph(f_1 \cap f_2)$.

Theorem 1. *For any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ time complexity of algorithm 1 is*

$$T = O((length(\mathsf{L}_{f_1}) + length(\mathsf{L}_{f_2}))(\log |V| + \log |A|)) \ (|V| \to \infty). \quad (8)$$

Proof. Let us estimate time complexity of steps of algorithm 1 under the assumption that we are shifting through the doubly linked lists L_{f_1} and L_{f_2}, from their beginnings to their ends, no more, than on one line item.

Time complexity for each of steps 1, and 2 is

$$T = O(1) \ (|V| \to \infty). \quad (9)$$

Time complexity for each of steps 3, 6, and 7 is

$$T = O(\log |V| + \log |A|) \ (|V| \to \infty). \quad (10)$$

Time complexity for each of steps 4, 5, and 8 is

$$T = O(\log |V|) \ (|V| \to \infty). \quad (11)$$

The number of cycles via operation of algorithm 1 doesn't exceed the value

$$length(\mathsf{L}_{f_1}) + length(\mathsf{L}_{f_2}) \quad (12)$$

Formulae (9)-(12) imply that formula (8) holds. □

Corollary 1.1. *For any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation for time complexity of algorithm 1 is true*

$$T = O(|V|(\log |V| + \log |A|)) \ (|V| \to \infty). \quad (13)$$

Proof. Substituting (7) in (8), we get that formula (13) holds. □

Corollary 1.2. *Let $|V| \to \infty$. If either $|A| = o(|V|)$, or $|A| = O(|V|)$, then for any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation of time complexity for algorithm 1 is true*

$$T = O(|V| \log |V|) \ (|V| \to \infty). \tag{14}$$

Proof. Suppose, that $|V| \to \infty$. Substituting either $|A| = o(|V|)$, or $|A| = O(|V|)$ in (13), we get that formula (14) holds. □

Corollary 1.3. *Let $|V| \to \infty$. If either $|V| = o(|A|)$, or $|V| = O(|A|)$, then for any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation for time complexity of algorithm 1 is true*

$$T = O(|A| \log |V|) \ (|V| \to \infty). \tag{15}$$

Proof. Suppose, that $|V| \to \infty$. Substituting either $|V| = o(|A|)$, or $|V| = O(|A|)$ in (14), we get that formula (15) holds. □

The operation $\setminus$ on the set $\mathfrak{F}_{V,A}$ can be implemented as follows.

Algorithm 2.
Input: doubly linked lists L_{f_1} and L_{f_2}.
Output: doubly linked list $\mathsf{L}_{f_1 \setminus f_2}$.
Step 1. $\mathsf{L}_{f_1 \setminus f_2} := \emptyset$, $j_3 := 0$.
Step 2. If $length(\mathsf{L}_{f_1}) = 0$, then HALT.
Step 3. If $length(\mathsf{L}_{f_2}) = 0$, then $\mathsf{L}_{f_1 \setminus f_2} := \mathsf{L}_{f_1}$, and HALT.
Step 4. $j_1 := 1$, $j_2 := 1$, $\mathbf{u}_1 := \mathsf{L}_{f_1}(i^{(1)}_{j_1})$, $\mathbf{u}_2 := \mathsf{L}_{f_2}(i^{(2)}_{j_2})$.
Step 5. If $\mathrm{pr}_2\mathbf{u}_1 = \mathrm{pr}_2\mathbf{u}_2$, then go to step 9.
Step 6. If $\mathrm{pr}_2\mathbf{u}_1 <_V \mathrm{pr}_2\mathbf{u}_2$,
then $r := 0$, and Call *Procedure2.1*(r).
Step 7. If $\mathrm{pr}_4\mathbf{u}_2 = *$, then $r := 1$, and Call *Procedure2.1*(r).
Step 8. $j_2 := j_2 + 1$, $\mathbf{u}_2 := \mathsf{L}_{f_2}(i^{(2)}_{j_2})$, and go to step 5.
Step 9. If $\mathrm{pr}_3\mathbf{u}_1 \neq \mathrm{pr}_3\mathbf{u}_2$,
then $r := 0$, and Call *Procedure2.1*(r).
Step 10. If $\mathrm{pr}_4\mathbf{u}_1 = *$, then HALT,
else $j_1 := j_1 + 1$, $\mathbf{u}_1 := \mathsf{L}_{f_1}(i^{(1)}_{j_1})$,
and go to step 7.

Procedure2.1(r)
begin;
M3: If $j_3 = 0$,
then $j_3 := j_3 + 1$, $\mathsf{L}_{f_1 \setminus f_2}(i^{(3)}_{j_3}) := (*, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$,
and go to M1;
M2: $\mathrm{pr}_4\mathsf{L}_{f_1 \setminus f_2}(i^{(3)}_{j_3}) := j_3 + 1$,
$\mathsf{L}_{f_1 \setminus f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$, $j_3 := j_3 + 1$;
M1: If $\mathrm{pr}_4\mathbf{u}_1 = *$, then HALT,
else $j_1 := j_1 + 1$, $\mathbf{u}_1 := \mathsf{L}_{f_1}(i^{(1)}_{j_1})$;
If $r = 0$, then go to step 7, else go to M3;
end;

Correctness of algorithm 2 is justified by the following two factors. Firstly, algorithm 2 always halts. Secondly, the list $\mathsf{L}_{f_1 \setminus f_2}$ consists of those and only those pairs $(v, a) \in V \times A$, which are elements of the set $graph(f_1 \setminus f_2)$.

Theorem 2. *For any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ time complexity of algorithm 2 is*

$$T = O((length(\mathsf{L}_{f_1}) + length(\mathsf{L}_{f_2}))(\log|V| + \log|A|)) \ (|V| \to \infty). \quad (16)$$

Proof is similar to proof of theorem 1.

Corollary 2.1. *For any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation for time complexity of algorithm 2 is true*

$$T = O(|V|(\log|V| + \log|A|)) \ (|V| \to \infty). \quad (17)$$

Proof is similar to proof of corollary 1.1.

Corollary 2.2. *Let $|V| \to \infty$. If either $|A| = o(|V|)$, or $|A| = O(|V|)$, then for any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation of time complexity for algorithm 2 is true*

$$T = O(|V| \log|V|) \ (|V| \to \infty). \quad (18)$$

Proof is similar to proof of corollary 1.2.

Corollary 2.3. *Let $|V| \to \infty$. If either $|V| = o(|A|)$, or $|V| = O(|A|)$, then for any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation for time complexity of algorithm 2 is true*

$$T = O(|A| \log |V|) \ (|V| \to \infty). \tag{19}$$

Proof is similar to proof of corollary 1.3.

The operation $\cup$ on the set $\mathfrak{F}_{V,A}$ can be implemented as follows.

Algorithm 3.

Input: doubly linked lists L_{f_1} and L_{f_2}.

Output: $\alpha \in \{0, 1\}$, where $\alpha = 0$, if operation $\cup$ is not defined for $f_1, f_2 \in \mathfrak{F}_{V,A}$, and $\alpha = 1$, if doubly linked list $\mathsf{L}_{f_1 \cup f_2}$ is computed.

Step 1. $\mathsf{L}_{f_1 \cup f_2} := \emptyset$, $\alpha := 0$, $j_3 := 0$.

Step 2. If $length(\mathsf{L}_{f_1}) = 0$,
then $\mathsf{L}_{f_1 \cup f_2} := \mathsf{L}_{f_2}$, $\alpha := 1$, and HALT.

Step 3. If $length(\mathsf{L}_{f_2}) = 0$,
then $\mathsf{L}_{f_1 \cup f_2} := \mathsf{L}_{f_1}$, $\alpha := 1$, and HALT.

Step 4. $j_1 := 1$, $j_2 := 1$, $\mathbf{u}_1 := \mathsf{L}_{f_1}(i^{(1)}_{j_1})$, $\mathbf{u}_2 := \mathsf{L}_{f_2}(i^{(2)}_{j_2})$.

Step 5. If $\mathrm{pr}_2\mathbf{u}_1 <_V \mathrm{pr}_2\mathbf{u}_2$,
then $r := 1$, and Call *Procedure3.1*(r).

Step 6. If $\mathrm{pr}_2\mathbf{u}_2 <_V \mathrm{pr}_2\mathbf{u}_1$,
then $r := 2$, and Call *Procedure3.1*(r).

Step 7. If $\mathrm{pr}_3\mathbf{u}_1 \neq \mathrm{pr}_3\mathbf{u}_2$, then $\alpha := 0$, and HALT.

Step 8. If $j_3 = 0$, then $j_3 := j_3 + 1$,
$\mathsf{L}_{f_1 \cup f_2}(i^{(3)}_{j_3}) := (*, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$,
and go to step 10.

Step 9. $\mathrm{pr}_4\mathsf{L}_{f_1 \cup f_2}(i^{(3)}_{j_3}) := j_3 + 1$,
$\mathsf{L}_{f_1 \cup f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$, $j_3 := j_3 + 1$.

Step 10. If $\mathrm{pr}_4\mathbf{u}_1 = *$ and $\mathrm{pr}_4\mathbf{u}_2 = *$, then $\alpha := 1$, and HALT.

Step 11. If $\mathrm{pr}_4\mathbf{u}_1 = *$ and $\mathrm{pr}_4\mathbf{u}_2 \neq *$, then go to step 14.

Step 12. If $\mathrm{pr}_4\mathbf{u}_1 \neq *$ and $\mathrm{pr}_4\mathbf{u}_2 = *$, then go to step 16.

Step 13. $j_1 := j_1 + 1$, $j_2 := j_2 + 1$,
$\mathbf{u}_1 := \mathsf{L}_{f_1}(i^{(1)}_{j_1})$, $\mathbf{u}_2 := \mathsf{L}_{f_2}(i^{(2)}_{j_2})$, and go to step 5.

Step 14. $j_2 := j_2 + 1$, $\mathbf{u}_2 := \mathsf{L}_{f_2}(i^{(2)}_{j_2})$, $\mathrm{pr}_4\mathsf{L}_{f_1 \cup f_2}(i^{(3)}_{j_3}) := j_3 + 1$,

$\mathsf{L}_{f_1\cup f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_2, \mathrm{pr}_3\mathbf{u}_2, *)$, $j_3 := j_3 + 1$.

Step 15. If $\mathrm{pr}_4\mathbf{u}_2 = *$, then $\alpha := 1$, and HALT, else go to step 14.

Step 16. $j_1 := j_1 + 1$, $\mathbf{u}_1 := \mathsf{L}_{f_1}(i^{(1)}_{j_1})$, $\mathrm{pr}_4\mathsf{L}_{f_1\cup f_2}(i^{(3)}_{j_3}) := j_3 + 1$,

$\mathsf{L}_{f_1\cup f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$, $j_3 := j_3 + 1$.

Step 17. If $\mathrm{pr}_4\mathbf{u}_1 = *$, then $\alpha := 1$, and HALT, else go to step 16.

Procedure3.1(r)

begin;

If $j_3 = 0$,

then $j_3 := j_3 + 1$, $\mathsf{L}_{f_1\cup f_2}(i^{(3)}_{j_3}) := (*, \mathrm{pr}_2\mathbf{u}_r, \mathrm{pr}_3\mathbf{u}_r, *)$,

and go to M1;

$\mathrm{pr}_4\mathsf{L}_{f_1\cup f_2}(i^{(3)}_{j_3}) := j_3 + 1$,

$\mathsf{L}_{f_1\cup f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_r, \mathrm{pr}_3\mathbf{u}_r, *)$, $j_3 := j_3 + 1$;

M1: If $\mathrm{pr}_4\mathbf{u}_r \neq *$,

then $j_r := j_r + 1$, $\mathbf{u}_r := \mathsf{L}_{f_r}(i^{(r)}_{j_r})$, and go to step 5.

$\mathsf{L}_{f_1\cup f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_{3-r}, \mathrm{pr}_3\mathbf{u}_{3-r}, *)$, $j_3 := j_3 + 1$;

M2: If $\mathrm{pr}_4\mathbf{u}_{3-r} = *$, then $\alpha := 1$, and HALT;

$j_{3-r} := j_{3-r}+1$, $\mathbf{u}_{3-r} := \mathsf{L}_{f_{3-r}}(i^{(3-r)}_{j_{3-r}})$, $\mathrm{pr}_4\mathsf{L}_{f_1\cup f_2}(i^{(3)}_{j_3}) := j_3+1$,

$\mathsf{L}_{f_1\cup f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_{3-r}, \mathrm{pr}_3\mathbf{u}_{3-r}, *)$, $j_3 := j_3+1$, go to M2;

end;

Correctness of algorithm 3 is justified by the following two factors. Firstly, algorithm 3 always halts. Secondly, $\alpha = 1$ if and only if operation $\cup$ is defined for $f_1, f_2 \in \mathfrak{F}_{V,A}$, and doubly linked list $\mathsf{L}_{f_1\cup f_2}$ is computed.

Theorem 3. *For any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ time complexity of algorithm 3 is*

$$T = O((length(\mathsf{L}_{f_1}) + length(\mathsf{L}_{f_2}))(\log|V| + \log|A|)) \ (|V| \to \infty). \quad (20)$$

Proof is similar to proof of theorem 1.

Corollary 3.1. *For any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation for time complexity of algorithm 3 is true*

$$T = O(|V|(\log|V| + \log|A|)) \ (|V| \to \infty). \quad (21)$$

Proof is similar to proof of corollary 1.1.

Corollary 3.2. *Let $|V| \to \infty$. If either $|A| = o(|V|)$, or $|A| = O(|V|)$, then for any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation of time complexity for algorithm 3 is true*

$$T = O(|V| \log |V|) \ (|V| \to \infty). \tag{22}$$

Proof is similar to proof of corollary 1.2.

Corollary 3.3. *Let $|V| \to \infty$. If either $|V| = o(|A|)$, or $|V| = O(|A|)$, then for any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation for time complexity of algorithm 3 is true*

$$T = O(|A| \log |V|) \ (|V| \to \infty). \tag{23}$$

Proof is similar to proof of corollary 1.3.

The operation $\oplus$ on the set $\mathfrak{F}_{V,A}$ can be implemented as follows.

Algorithm 4.

Input: doubly linked lists L_{f_1} and L_{f_2}.

Output: $\alpha \in \{0, 1\}$, where $\alpha = 0$, if operation $\oplus$ is not defined for $f_1, f_2 \in \mathfrak{F}_{V,A}$, and $\alpha = 1$, if doubly linked list $\mathsf{L}_{f_1 \oplus f_2}$ is computed.

Step 1. $\mathsf{L}_{f_1 \oplus f_2} := \emptyset$, $\alpha := 0$, $j_3 := 0$.

Step 2. If $length(\mathsf{L}_{f_1}) = 0$, then $\mathsf{L}_{f_1 \oplus f_2} := \mathsf{L}_{f_2}$, $\alpha := 1$, and HALT.

Step 3. If $length(\mathsf{L}_{f_2}) = 0$, then $\mathsf{L}_{f_1 \oplus f_2} := \mathsf{L}_{f_1}$, $\alpha := 1$, and HALT.

Step 4. $j_1 := 1$, $j_2 := 1$, $\mathbf{u}_1 := \mathsf{L}_{f_1}(i^{(1)}_{j_1})$, $\mathbf{u}_2 := \mathsf{L}_{f_2}(i^{(2)}_{j_2})$.

Step 5. If $\mathrm{pr}_2\mathbf{u}_1 <_V \mathrm{pr}_2\mathbf{u}_2$,
then $r := 1$, and Call *Procedure4.1*(r).

Step 6. If $\mathrm{pr}_2\mathbf{u}_2 <_V \mathrm{pr}_2\mathbf{u}_1$,
then $r := 2$, and Call *Procedure4.1*(r).

Step 7. If $\mathrm{pr}_3\mathbf{u}_1 \neq \mathrm{pr}_3\mathbf{u}_2$, then $\alpha := 0$, and HALT.

Step 8. If $\mathrm{pr}_4\mathbf{u}_1 = *$ and $\mathrm{pr}_4\mathbf{u}_2 = *$, then $\alpha := 1$, and HALT.

Step 9. If $\mathrm{pr}_4\mathbf{u}_1 = *$ and $\mathrm{pr}_4\mathbf{u}_2 \neq *$,
then $r := 2$ and Call *Procedure4.3*(r).

Step 10. If $\mathrm{pr}_4\mathbf{u}_1 \neq *$ and $\mathrm{pr}_4\mathbf{u}_2 = *$,
then $r := 1$ and Call *Procedure4.3*(r).

Step 11. $j_1 := j_1 + 1$, $j_2 := j_2 + 1$,
$\mathbf{u}_1 := \mathsf{L}_{f_1}(i^{(1)}_{j_1})$, $\mathbf{u}_2 := \mathsf{L}_{f_2}(i^{(2)}_{j_2})$, and go to step 5.

Procedure4.1(r)
begin;
If $j_3 = 0$, then $j_3 := j_3 + 1$,
$\mathsf{L}_{f_1 \oplus f_2}(i^{(3)}_{j_3}) := (*, \mathrm{pr}_2\mathbf{u}_r, \mathrm{pr}_3\mathbf{u}_r, *)$,
and go to M1;
$\mathrm{pr}_4\mathsf{L}_{f_1 \oplus f_2}(i^{(3)}_{j_3}) := j_3 + 1$,
$\mathsf{L}_{f_1 \oplus f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_r, \mathrm{pr}_3\mathbf{u}_r, *)$, $j_3 := j_3 + 1$;
M1: If $\mathrm{pr}_4\mathbf{u}_r \neq *$,
then $j_1 := j_1 + 1$, $\mathbf{u}_r := \mathsf{L}_{f_r}(i^{(r)}_{j_r})$, and go to step 5;
$\mathsf{L}_{f_1 \oplus f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_{3-r}, \mathrm{pr}_3\mathbf{u}_{3-r}, *)$, $j_3 := j_3 + 1$;
Call *Procedure4.2*$(3 - r)$;
end;

Procedure4.2(r)
begin;
M1: If $\mathrm{pr}_4\mathbf{u}_r = *$, then $\alpha := 1$, and HALT;
$j_r := j_r + 1$, $\mathbf{u}_r := \mathsf{L}_{f_r}(i^{(r)}_{j_r})$;
If $j_3 = 0$,
then $j_3 := j_3 + 1$, $\mathsf{L}_{f_1 \oplus f_2}(i^{(3)}_{j_3}) := (*, \mathrm{pr}_2\mathbf{u}_r, \mathrm{pr}_3\mathbf{u}_r, *)$,
and go to M1;
$\mathrm{pr}_4\mathsf{L}_{f_1 \oplus f_2}(i^{(3)}_{j_3}) := j_3 + 1$, $\mathsf{L}_{f_1 \oplus f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_r, \mathrm{pr}_3\mathbf{u}_r, *)$,
$j_3 := j_3 + 1$, and go to M1;
end;

Procedure4.3(r)
begin;
M1: $j_r := j_r + 1$, $\mathbf{u}_r := \mathsf{L}_{f_r}(i^{(r)}_{j_r})$;
If $j_3 = 0$,
then $j_3 := j_3 + 1$, $\mathsf{L}_{f_1 \oplus f_2}(i^{(3)}_{j_3}) := (*, \mathrm{pr}_2\mathbf{u}_r, \mathrm{pr}_3\mathbf{u}_r, *)$,
and go to M2;
$\mathrm{pr}_4\mathsf{L}_{f_1 \oplus f_2}(i^{(3)}_{j_3}) := j_3 + 1$,
$\mathsf{L}_{f_1 \oplus f_2}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_r, \mathrm{pr}_3\mathbf{u}_r, *)$, $j_3 := j_3 + 1$;

M2: If $\mathrm{pr}_4\mathbf{u}_2 = *$, then $\alpha := 1$, and HALT, else go to M1;
end;

Correctness of algorithm 4 is justified by the following two factors. Firstly, algorithm 4 always halts. Secondly, $\alpha = 1$ if and only if operation $\oplus$ is defined for $f_1, f_2 \in \mathfrak{F}_{V,A}$, and doubly linked list $\mathsf{L}_{f_1 \oplus f_2}$ is computed.

Theorem 4. *For any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ time complexity of algorithm 4 is*

$$T = O((length(\mathsf{L}_{f_1}) + length(\mathsf{L}_{f_2}))(\log|V| + \log|A|)) \ (|V| \to \infty). \quad (24)$$

Proof is similar to proof of theorem 1.

Corollary 4.1. *For any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation for time complexity of algorithm 4 is true*

$$T = O(|V|(\log|V| + \log|A|)) \ (|V| \to \infty). \quad (25)$$

Proof is similar to proof of corollary 1.1.

Corollary 4.2. *Let $|V| \to \infty$. If either $|A| = o(|V|)$, or $|A| = O(|V|)$, then for any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation of time complexity for algorithm 4 is true*

$$T = O(|V| \log|V|) \ (|V| \to \infty). \quad (26)$$

Proof is similar to proof of corollary 1.2.

Corollary 4.3. *Let $|V| \to \infty$. If either $|V| = o(|A|)$, or $|V| = O(|A|)$, then for any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation for time complexity of algorithm 4 is true*

$$T = O(|A| \log|V|) \ (|V| \to \infty). \quad (27)$$

Proof is similar to proof of corollary 1.3.

It is evident that to compute the result of each of operations, $\triangleright$ and $\boxplus$, defined by formulae (5) and (6), it is needed some algorithm, which for any given nominative sets $f, g \in \mathfrak{F}_{V,A}$, presented by doubly linked

lists L_f and L_g, computes the doubly linked list $\mathsf{L}_{f|_{Dom\ f \setminus Dom\ g}}$. Using this algorithm, we can apply the algorithm 3 to compute the result of each of operations $\triangleright$ and $\boxplus$.

An algorithm, which for any given nominative sets $f, g \in \mathfrak{F}_{V,A}$, presented by doubly linked lists L_f and L_g, computes the doubly linked list $\mathsf{L}_{f|_{Dom\ f \setminus Dom\ g}}$, can be designed as follows.

ALGORITHM 5.
Input: doubly linked lists L_f and L_g.
Output: doubly linked list $\mathsf{L}_{f|_{Dom\ f \setminus Dom\ g}}$.
Step 1. $\mathsf{L}_{f|_{Dom\ f \setminus Dom\ g}} := \emptyset$, $j_3 := 0$.
Step 2. If $length(\mathsf{L}_f) = 0$, then HALT.
Step 3. If $length(\mathsf{L}_g) = 0$, then $\mathsf{L}_{f|_{Dom\ f \setminus Dom\ g}} := \mathsf{L}_f$, and HALT.
Step 4. $j_1 := 1$, $j_2 := 1$, $\mathbf{u}_1 := \mathsf{L}_f(i^{(1)}_{j_1})$, $\mathbf{u}_2 := \mathsf{L}_g(i^{(2)}_{j_2})$.
Step 5. If $\mathrm{pr}_2\mathbf{u}_1 <_V \mathrm{pr}_2\mathbf{u}_2$, then Call *Procedure5.1.*
Step 6. If $\mathrm{pr}_2\mathbf{u}_2 <_V \mathrm{pr}_2\mathbf{u}_1$, then Call *Procedure5.2.*
Step 7. If $\mathrm{pr}_4\mathbf{u}_1 = *$,
then HALT,
else $j_1 := j_1 + 1$, $\mathbf{u}_1 := \mathsf{L}_f(i^{(1)}_{j_1})$, and go to step 5.

Procedure5.1.
begin;
If $j_3 = 0$ then $j_3 := j_3 + 1$,
$\mathsf{L}_{f|_{Dom\ f \setminus Dom\ g}}(i^{(3)}_{j_3}) := (*, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$,
and go to M1;
$\mathrm{pr}_4\mathsf{L}_{f|_{Dom\ f \setminus Dom\ g}}(i^{(3)}_{j_3}) := j_3 + 1$,
$\mathsf{L}_{f|_{Dom\ f \setminus Dom\ g}}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$, $j_3 := j_3 + 1$;
M1: If $\mathrm{pr}_4\mathbf{u}_1 \neq *$,
then $j_1 := j_1 + 1$, $\mathbf{u}_1 := \mathsf{L}_f(i^{(1)}_{j_1})$, and go to step 5,
else HALT;
end;

Procedure5.2.
begin;
If $\mathrm{pr}_4\mathbf{u}_2 \neq *$,

then $j_2 := j_2 + 1$, $\mathbf{u}_2 := \mathsf{L}_g(i^{(2)}_{j_2})$, and go to step 5;
M1: If $j_3 = 0$ then $j_3 := j_3 + 1$,
$\mathsf{L}_{f|_{Dom\, f \setminus Dom\, g}}(i^{(3)}_{j_3}) := (*, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$,
and go to M2;
$\mathrm{pr}_4\mathsf{L}_{f|_{Dom\, f \setminus Dom\, g}}(i^{(3)}_{j_3}) := j_3 + 1$,
$\mathsf{L}_{f|_{Dom\, f \setminus Dom\, g}}(i^{(3)}_{j_3+1}) := (j_3, \mathrm{pr}_2\mathbf{u}_1, \mathrm{pr}_3\mathbf{u}_1, *)$, $j_3 := j_3 + 1$;
M2: If $\mathrm{pr}_4\mathbf{u}_1 \neq *$,
then $j_1 := j_1 + 1$, $\mathbf{u}_1 := \mathsf{L}_f(i^{(1)}_{j_1})$, and go to M1,
else HALT;
end;

Correctness of algorithm 5 is justified by the following two factors. Firstly, algorithm 5 always halts. Secondly, the list $\mathsf{L}_{f|_{Dom\, f \setminus Dom\, g}}$ consists of those and only those pairs $(v, a) \in V \times A$, which are elements of the set $graph(f|_{Dom\, f \setminus Dom\, g})$.

Theorem 5. *For any nominative sets $f, g \in \mathfrak{F}_{V,A}$ time complexity of algorithm 5 is*

$$T = O((length(\mathsf{L}_f) + length(\mathsf{L}_g))(\log|V| + \log|A|)) \ (|V| \to \infty). \quad (28)$$

Proof is similar to proof of theorem 1.

Corollary 5.1. *For any nominative sets $f, g \in \mathfrak{F}_{V,A}$ the following estimation for time complexity of algorithm 5 is true*

$$T = O(|V|(\log|V| + \log|A|)) \ (|V| \to \infty). \quad (29)$$

Proof is similar to proof of corollary 1.1.

Corollary 5.2. *Let $|V| \to \infty$. If either $|A| = o(|V|)$, or $|A| = O(|V|)$, then for any nominative sets $f, g \in \mathfrak{F}_{V,A}$ the following estimation of time complexity for algorithm 5 is true*

$$T = O(|V|\log|V|) \ (|V| \to \infty). \quad (30)$$

Proof is similar to proof of corollary 1.2.

Corollary 5.3. *Let $|V| \to \infty$. If either $|V| = o(|A|)$, or $|V| = O(|A|)$, then for any nominative sets $f, g \in \mathfrak{F}_{V,A}$ the following estimation for time complexity of algorithm 5 is true*

$$T = O(|A| \log |V|) \ (|V| \to \infty). \tag{31}$$

Proof is similar to proof of corollary 1.3.

The operations $\triangleright$ and $\boxplus$ on the set $\mathfrak{F}_{V,A}$ can be implemented as follows.

ALGORITHM 6.
Input: doubly linked lists L_{f_1} and L_{f_2}.
Output: doubly linked list $\mathsf{L}_{f_1 \triangleright f_2}$.
Step 1. Applying algorithm 5 to doubly linked lists L_{f_2} and L_{f_1}, we design doubly linked list $\mathsf{L}_{f_2|_{Dom\ f_2 \setminus Dom\ f_1}}$.
Step 2. Applying algorithm 3 to doubly linked lists L_{f_1} and $\mathsf{L}_{f_2|_{Dom\ f_2 \setminus Dom\ f_1}}$, we design doubly linked list $\mathsf{L}_{f_1 \triangleright f_2}$.

ALGORITHM 7.
Input: doubly linked lists L_{f_1} and L_{f_2}.
Output: doubly linked list $\mathsf{L}_{f_1 \boxplus f_2}$.
Step 1. Applying algorithm 5 to doubly linked lists L_{f_1} and L_{f_2}, we design doubly linked list $\mathsf{L}_{f_1|_{Dom\ f_1 \setminus Dom\ f_2}}$.
Step 2. Applying algorithm 5 to doubly linked lists L_{f_2} and L_{f_1}, we design doubly linked list $\mathsf{L}_{f_2|_{Dom\ f_2 \setminus Dom\ f_1}}$.
Step 3. Applying algorithm 3 to doubly linked lists $\mathsf{L}_{f_1|_{Dom\ f_1 \setminus Dom\ f_2}}$ and $\mathsf{L}_{f_2|_{Dom\ f_2 \setminus Dom\ f_1}}$, we design doubly linked list $\mathsf{L}_{f_1 \boxplus f_2}$.

Estimations for time complexity of algorithms 3 and 5 (i.e. theorems 3 and 5, and corresponding corollaries) imply that the following estimations for time complexity of algorithms 6 and 7 hold.

Theorem 6. *For any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ time complexity for each of algorithms 6 and 7 is*

$$T = O((length(\mathsf{L}_{f_1}) + length(\mathsf{L}_{f_2}))(\log |V| + \log |A|)) \ (|V| \to \infty). \tag{32}$$

Corollary 6.1. *For any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation of time complexity for each of algorithms 6 and 7 is true*

$$T = O(|V|(\log|V| + \log|A|)) \ (|V| \to \infty). \tag{33}$$

Corollary 6.2. *Let $|V| \to \infty$. If either $|A| = o(|V|)$, or $|A| = O(|V|)$, then for any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation of time complexity for each of algorithms 6 and 7 is true*

$$T = O(|V| \log|V|) \ (|V| \to \infty). \tag{34}$$

Corollary 6.3. *Let $|V| \to \infty$. If either $|V| = o(|A|)$, or $|V| = O(|A|)$, then for any nominative sets $f_1, f_2 \in \mathfrak{F}_{V,A}$ the following estimation of time complexity for each of algorithms 6 and 7 is true*

$$T = O(|A| \log|V|) \ (|V| \to \infty). \tag{35}$$

The obtained results justify the factor, that in the case of linear ordering on the set of names, all basic set-theoretic operations over nominative sets, presented by doubly linked lists with ordering of names in increasing strength, can be implemented by fast algorithms. The same is also true for operations of inserting elements in any nominative set, and of deleting elements from any nominative set.

Above, it has been investigated the case, when data are elements of an abstract set. Obviously that these results can be easily elaborated in detail for any case, when this or that structure is defined on the set of data. One of such examples will be considered in the next section.

4 Applications to table algebra

Relational databases are widely used in modern software systems. It is well known, that any relational database deals, in essence, with finite relations, defined on some Cartesian products [17].

Mathematical model, intended for developing and theoretic analysis of relational databases, as well as of associated query languages, is some table algebra. It can be characterized as follows.

For any relation its scheme is defined, which is a set of attributes. Any line of a relation is defined as a set of ordered pairs (attribute, value of attribute), where attribute transverses all values according to the scheme of the relation. A relation itself is defined as a set of lines. Proceeding from this set-theoretic representation, formal theory of relations can be implemented easily into table algebra. Thus, analysis in detail of set-theoretic operations over tables is essential for any table algebra.

Unfortunately, in investigation of table algebras all efforts are bent on development of descriptive theory, while there are practically no researches devoted to algorithms elaboration and analysis of their complexity.

Due to this factor, it is worth to point the paper [18], where there have been investigated worst-case and average-case time complexity of algorithms implemented for execution of three main set-theoretic operations over tables, namely: intersection, union and difference.

It is worth to note that in [18] time complexity is considered as the number of elementary steps. As the result, all estimations, established in [18], are typical estimations, that can be established for set-theoretic operations on abstract sets.

Let us consider, how the results, established in Section 3, can be applied effectively to the analysis of complexity of set-theoretic operations in table algebras.

Let $\mathcal{A} = \{A_i | i = 1, \ldots, k\}$ be the set of all attributes that are used in the given table algebra. The active domain of an attribute A_i $(i = 1, \ldots, k)$ is denoted by $Dom\, A_i$. It is evident that any set $Dom\, A_i$ $(i = 1, \ldots, k)$ can be linearly ordered. Due to this factor, we fix some linear ordering $<_{A_i}$ on each set $Dom\, A_i$ $(i = 1, \ldots, k)$. Thus, each Cartesian product $\mathcal{D}_{i_1,\ldots,i_l} = Dom\, A_{i_1} \times \cdots \times Dom\, A_{i_l}$, where $i_j \in \{1, \ldots, k\}$ for all $j \in \{1, \ldots, l\}$, is linearly ordered by lexicographic order $\prec_{i_1,\ldots,i_l}$.

The following two approaches, based on the theory of nominative sets with linearly ordered set of names, can be applied in table algebras for implementing set-theoretic operations on the set of relations of the form $\rho \subseteq \mathcal{D}_{i_1,\ldots,i_l}$, where $i_1, \ldots, i_l \in \{1, \ldots, k\}$ are fixed integers.

Approach I. It is supposed that the following two assumptions hold:

1) the set of all nominative sets is $\mathfrak{F}_{\mathcal{D}_{i_1,\ldots,i_l},\{1\}}$;

2) each nominative set $f_\rho \in \mathfrak{F}_{\mathcal{D}_{i_1,\ldots,i_l},\{1\}}$ is presented by doubly linked list L_{f_ρ} with ordering of names in the increasing strength.

Under these assumptions all algorithms, designed in Chapter 3, can be applied directly for implementation in table algebras operations $\cap$, $\setminus$, $\cup$, $\oplus$, $\triangleright$, and $\boxplus$, defined by formulae (1)-(6). Moreover, the following theorem holds

Theorem 7. *For any nominative sets $f_{\rho_1}, f_{\rho_2} \in \mathfrak{F}_{\mathcal{D}_{i_1,\ldots,i_l},\{1\}}$, presented by doubly linked lists with ordering of names in the increasing strength, for each of algorithms 1-7, implemented for execution of operations $\cap$, $\setminus$, $\cup$, $\oplus$, $\triangleright$, and $\boxplus$, defined by formulae (1)-(6), the worst-case time complexity under logarithmic weight is*

$$T = O((|\rho_1| + |\rho_2|) \log \prod_{j=1}^{l} |Dom\, A_{i_j}|) \ \ (|V| \to \infty). \tag{36}$$

It is worth to point out that in the considered case operations $\cup$ and $\oplus$ are usual total set-theoretic operations on the set $\mathfrak{F}_{\mathcal{D}_{i_1,\ldots,i_l},\{1\}}$.

Approach II. It is supposed that the following three assumptions hold:

1) the set of all nominative sets is $\mathfrak{F}_{V,\mathcal{D}_{i_1,\ldots,i_l}}$, where V is some set of names, ordered by linear ordering relation $<_V$;

2) each relation $\rho \subseteq \mathcal{D}_{i_1,\ldots,i_l}$ is presented by the nominative set $f_\rho \in \mathfrak{F}_{V,\mathcal{D}_{i_1,\ldots,i_l}}$, such that the following condition holds:

$$(v_1, (a_{i_1}^{(1)}, \ldots, a_{i_l}^{(1)})), (v_2, (a_{i_1}^{(2)}, \ldots, a_{i_l}^{(2)})) \in f_\rho \,\&\, v_1 <_V v_2 \Rightarrow$$
$$\Rightarrow (a_{i_1}^{(1)}, \ldots, a_{i_l}^{(1)}) \preceq_{i_1,\ldots,i_l} (a_{i_1}^{(2)}, \ldots, a_{i_l}^{(2)}),$$

i.e. each nominative set $f_\rho \in \mathfrak{F}_{V,\mathcal{D}_{i_1,\ldots,i_l}}$ is a nondecreasing (possibly partial) mapping from V to $\mathcal{D}_{i_1,\ldots,i_l}$;

3) each nominative set $f_\rho \in \mathfrak{F}_{V,\mathcal{D}_{i_1,\ldots,i_l}}$ is presented by doubly linked list L_{f_ρ} with ordering of names in the increasing strength.

Under these assumptions all algorithms, designed in Chapter 3, can be applied directly for implementation in table algebras operations $\cap$,

$\setminus$, $\cup$, $\oplus$, $\triangleright$, and $\boxplus$, defined by formulae (1)-(6). Moreover, the following theorem holds

Theorem 8. *For any nominative sets $f_{\rho_1}, f_{\rho_2} \in \mathfrak{F}_{V,\mathcal{D}_{i_1,\ldots,i_l}}$, presented by doubly linked lists with ordering of names in the increasing strength, for each of algorithms 1-7, implemented for execution of operations $\cap$, $\setminus$, $\cup$, $\oplus$, $\triangleright$, and $\boxplus$, defined by formulae (1)-(6), the worst-case time complexity under logarithmic weight is*

$$T = O((|\rho_1| + |\rho_2|)(\log |V| + \log \prod_{j=1}^{l} |Dom\, A_{i_j}|)) \ (|V| \to \infty). \quad (37)$$

It is worth to note that in the considered case operations $\cup$ and $\oplus$ are partial operations on the set $\mathfrak{F}_{V,\mathcal{D}_{i_1,\ldots,i_l}}$. For table algebras this factor means that some of these or the others additional conditions are associated with the set V of names.

Besides, if we take into account the factor, that each nominative set $f_\rho \in \mathfrak{F}_{V,\mathcal{D}_{i_1,\ldots,i_l}}$ is a nondecreasing (possibly partial) mapping from V to $\mathcal{D}_{i_1,\ldots,i_l}$, then we can speed-up execution of algorithms 1-7.

5 Conclusions

In the given paper it has been formed a strong base, sufficient for effective implementation of application-oriented algorithms theory in theory of nominative sets. Developing these results for hierarchical types of data, multidimensional arrays, lists, trees, algebraic data types, etc. forms some trend for future research.

References

[1] T. Hoare, "The verifying compiler: A grand challenge for computing research," in *Modular Programming Languages. JMLC 2003* (Lecture Notes in Computer Science, vol 2789), L. Boszormenyi, P. Schojer, Eds., Berlin, Germany: Springer, 2003, pp. 25–35.

[2] R. Floyd, "Assigning meanings to programs," in *In Proceedings of a Symposium on Applied Mathematics,* vol. 19, 1967, pp. 19–31.

[3] C.A.R. Hoare, "An axiomatic basis for computer programming," *Communications of the ACM*, vol. 12, no. 10, pp. 576–580, 1969.

[4] M.A. Jackson, *Principles of program design*, London, UK: Academic Press, 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, no. 8, pp. 613–641, 1978.

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

[7] H.R. Nielson and F. Nielson, *Semantics with applications – a formal introduction* (Wiley professional computing), John Wiley & Sons Inc, 1992, 252p. ISBN-10: 0471929808. ISBN-13: 978-0471929802.

[8] J. Woodcock, P.G. Larsen, J. Bicarregui and J. Fitzgerald, "Formal methods: practice and experience," *ACM Computing Surveys*, vol. 41, no. 4, Article No. 19, pp. 1–36, 2009. DOI: 10.1145/1592434.1592436.

[9] C.A.R. Hoare, J. Misra, G. Leavens, and N. Shankar, "The verified software initiative: A manifesto," *ACM Computing Surveys*, vol. 41, no. 4, Article No. 22, pp. 1–8, 2009. DOI: 10.1145/1592434.1592439.

[10] A.V. Lamsweerde, *Requirements engineering: from system goals to UML models to software specifications*, Wiley Publishing, 2009, 712 p. ISBN: 978-0-470-01270-3.

[11] D. Sannella and A. Tarlecki, *Foundations of Algebraic Specification and Formal Software Development* (Monographs in Theoretical Computer Science. An EATCS Series), Springer-Verlag Berlin Heidelberg, 2012, XVI+584 p. ISBN: 978-3-642-17335-6.

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

[13] M.S. Nikitchenko and S.S. Shkilnjak, *Mathematical logic and algorithms theory*, Ukraine: Kiev National University Press, 2008, 528 p. (in Ukrainian). ISBN: 966-439-007-0.

[14] M.S. Nikitchenko and S.S. Shkilnjak, *Applied logic*, Ukraine: Kiev National University Press, 2013, 277 p. (in Ukrainian).

[15] V.G. Skobelev, M. Nikitchenko and Ie. Ivanov, "Set-theoretic analysis of nominative data," *Computer Science Journal of Moldova*, vol. 23, no. 3, pp. 270–288, 2015.

[16] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The design and analysis of computer algorithms*, Boston, MA, USA: Addison-Wesley, 1975.

[17] E.F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1969.

[18] V.N. Red'ko, D.B. Buy, I.S. Kanarskaya and A.S. Senchenko, "Precise estimates of the time complexity of implementing the algorithms of set-theoretic operations in table algebra," *Cybernetics and Systems Analysis*, vol. 53, no. 1, pp. 1–11, 2017.

Volodymyr G. Skobelev, Ievgen Ivanov,
Mykola Nikitchenko,

Received June 21, 2017

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@gmail.com

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