Computer Science Journal of Moldova, vol.**15**, no.1(43), 2007

# An Example of $\mathbf{\Pi^0_3}$-complete Infinitary Rational Relation

Olivier Finkel

**Abstract**

We give in this paper an example of infinitary rational relation, accepted by a 2-tape Büchi automaton, which is $\mathbf{\Pi^0_3}$-complete in the Borel hierarchy. Moreover the example of infinitary rational relation given in this paper has a very simple structure and can be easily described by its sections.
**Keywords:** infinitary rational relations; topological properties; Borel hierarchy; $\mathbf{\Pi^0_3}$-complete set.

## 1 Introduction

Acceptance of infinite words by finite automata was firstly considered in the sixties by Büchi in order to study decidability of the monadic second order theory of one successor over the integers [5]. Then the so called $\omega$-regular languages have been intensively studied and many applications have been found, see [33, 30, 25] for many results and references.

Since then many extensions of $\omega$-regular languages have been investigated as the classes of $\omega$-languages accepted by pushdown automata, Petri nets, Turing machines, see [33, 9, 30] for a survey of this work.

On the other side rational relations on finite words were studied in the sixties and played a fundamental role in the study of families of context free languages [4]. Investigations on their extension to rational relations on infinite words were carried out or mentioned in the books [1, 23]. Gire and Nivat studied infinitary rational relations in [16, 18]. Infinitary rational relations are subsets of $\Sigma_1^\omega \times \Sigma_2^\omega \times \ldots \times \Sigma_n^\omega$, where

$n$ is an integer $\geq 2$ and $\Sigma_1$, $\Sigma_2$, ... $\Sigma_n$ are finite alphabets, which are accepted by $n$-tape finite Büchi automata with $n$ asynchronous reading heads. So the class of infinitary rational relations extends both the class of finitary rational relations **and** the class of $\omega$-regular languages.

They have been much studied, in particular in connection with the rational functions they may define, see for example [8, 3, 28, 30, 27] for many results and references.

Notice that a rational relation $R \subseteq \Sigma_1^\omega \times \Sigma_2^\omega \times \ldots \times \Sigma_n^\omega$ may be seen as an $\omega$-language over the product alphabet $\Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_n$.

A way to study the complexity of languages of infinite words accepted by finite machines is to study their topological complexity and firstly to locate them with regard to the Borel and the projective hierarchies. This work is analysed for example in [29, 33, 9, 22, 30]. It is well known that every $\omega$-language accepted by a Turing machine with a Büchi or Muller acceptance condition is an analytic set and that $\omega$-regular languages are boolean combinations of $\mathbf{\Pi_2^0}$-sets hence $\mathbf{\Delta_3^0}$-sets, [30, 25].

The question of the topological complexity of relations on infinite words also naturally arises and was asked by Simonnet in [28]. It was also posed in a more general form by Lescow and Thomas in [22] (for infinite labelled partial orders) and in [32] where Thomas suggested to study reducibility notions and associated completeness results.

Every infinitary rational relation is an analytic set. We showed in [11] that there exist some infinitary rational relations which are analytic but non Borel sets. Considering Borel infinitary rational relations we prove in this paper that there exist some infinitary rational relations, accepted by 2-tape Büchi automata, which are $\mathbf{\Pi_3^0}$-complete.

Examples of $\mathbf{\Sigma_3^0}$-complete and $\mathbf{\Pi_3^0}$-complete infinitary rational relations have already been given in the conference paper [13]. But the proof of the existence of $\mathbf{\Pi_3^0}$-complete infinitary rational relations was only sketched and we used a coding of $\omega^2$-words by pairs of infinite words. We use in this paper a different coding of $\omega^2$-words. This way we get some infinitary rational relations which have a very simple structure and can be easily described by their sections.

The result given in this paper has two interests: 1) It gives a com-

plete proof of a result of [13]. 2) Some new ideas are here introduced with a new coding of $\omega^2$-words. Some of these new ideas led us further to the proof of very surprising results, answering the long standing open questions of the topological complexity of context free $\omega$-languages and of infinitary rational relations. In particular infinitary rational relations have the same topological complexity as $\omega$-languages accepted by Büchi Turing machines [14, 15] and for every recursive ordinal $\alpha$ there exist some $\mathbf{\Pi^0_\alpha}$-complete and some $\mathbf{\Sigma^0_\alpha}$-complete infinitary rational relations.

The result presented in this paper is still interesting, although the result of the paper [15] is stronger; we use here a coding of $\omega^2$-words while in [14, 15] we used a simulation of Turing machines and the examples of infinitary rational relations we obtain are different.

The result of this paper may also be compared with examples of $\mathbf{\Sigma^0_3}$-complete $\omega$-languages accepted by deterministic pushdown automata with the acceptance condition: "some stack content appears infinitely often during an infinite run", given by Cachat, Duparc, and Thomas in [6] or with examples of $\mathbf{\Sigma^0_n}$-complete and $\mathbf{\Pi^0_n}$-complete $\omega$-languages, $n \geq 1$, accepted by non-deterministic pushdown automata with Büchi acceptance condition given in [10].

The paper is organized as follows. In section 2 we introduce the notion of infinitary rational relations. In section 3 we recall definitions of Borel sets, and we prove our main result in section 4.

## 2 Infinitary rational relations

Let $\Sigma$ be a finite alphabet whose elements are called letters. A non-empty finite word over $\Sigma$ is a finite sequence of letters: $x = a_1 a_2 \dots a_n$ where for all integers $i \in [1; n]$ $a_i \in \Sigma$. We shall denote $x(i) = a_i$ the $i^{th}$ letter of $x$ and $x[i] = x(1) \dots x(i)$ for $i \leq n$. The length of $x$ is $|x| = n$. The empty word will be denoted by $\lambda$ and has 0 letter. Its length is 0. The set of finite words over $\Sigma$ is denoted $\Sigma^\star$. A (finitary) language $L$ over $\Sigma$ is a subset of $\Sigma^\star$. The usual concatenation product of $u$ and $v$ will be denoted by $u.v$ or just $uv$. For $V \subseteq \Sigma^\star$, we denote $V^\star = \{v_1 \dots v_n \mid \quad \forall i \in [1; n] \quad v_i \in V \quad \} \cup \{\lambda\}$.

The first infinite ordinal is $\omega$. An $\omega$-word over $\Sigma$ is an $\omega$ -sequence

$a_1 a_2 \ldots a_n \ldots$, where for all integers $i \geq 1$ $a_i \in \Sigma$. When $\sigma$ is an $\omega$-word over $\Sigma$, we write $\sigma = \sigma(1)\sigma(2)\ldots\sigma(n)\ldots$ and $\sigma[n] = \sigma(1)\sigma(2)\ldots\sigma(n)$ the finite word of length $n$, prefix of $\sigma$. The set of $\omega$-words over the alphabet $\Sigma$ is denoted by $\Sigma^\omega$. An $\omega$-language over an alphabet $\Sigma$ is a subset of $\Sigma^\omega$. For $V \subseteq \Sigma^\star$, $V^\omega = \{\sigma = u_1 \ldots u_n \ldots \in \Sigma^\omega \mid \forall i \geq 1 \ \ u_i \in V\}$ is the $\omega$-power of $V$. The concatenation product is extended to the product of a finite word $u$ and an $\omega$-word $v$: the infinite word $u.v$ is then the $\omega$-word such that: $(u.v)(k) = u(k)$ if $k \leq |u|$ , and $(u.v)(k) = v(k - |u|)$ if $k > |u|$.

If $A$ is a subset of $B$ we shall denote $A^- = B - A$ the complement of $A$ (in $B$).

We assume the reader to be familiar with the theory of formal languages and of $\omega$-regular languages. We recall that $\omega$-regular languages form the class of $\omega$-languages accepted by finite automata with a Büchi acceptance condition and this class is the omega Kleene closure of the class of regular finitary languages.

We are going now to introduce the notion of infinitary rational relation $R \subseteq \Sigma_1^\omega \times \Sigma_2^\omega$ via acceptance by 2-tape Büchi automata.

**Definition 2.1** *A 2-tape Büchi automaton is a 7-tuple $\mathcal{T} = (K, \Sigma_1, \Sigma_2, \Delta, q_0, F)$, where $K$ is a finite set of states, $\Sigma_1$, $\Sigma_2$, are finite alphabets, $\Delta$ is a finite subset of $K \times \Sigma_1^\star \times \Sigma_2^\star \times K$ called the set of transitions, $q_0$ is the initial state, and $F \subseteq K$ is the set of accepting states.*

*A computation $\mathcal{C}$ of the 2-tape Büchi automaton $\mathcal{T}$ over the pair $(u, v) \in \Sigma_1^\omega \times \Sigma_2^\omega$ is an infinite sequence of transitions*

$$(q_0, u_1, v_1, q_1), (q_1, u_2, v_2, q_2), \ldots (q_{i-1}, u_i, v_i, q_i), (q_i, u_{i+1}, v_{i+1}, q_{i+1}), \ldots$$

*such that: $u = u_1.u_2.u_3 \ldots$ and $v = v_1.v_2.v_3 \ldots$.*

*The computation is said to be successful iff there exists an accepting state $q_f \in F$ and infinitely many integers $i \geq 0$ such that $q_i = q_f$.*

*The infinitary rational relation $R(\mathcal{T}) \subseteq \Sigma_1^\omega \times \Sigma_2^\omega$ accepted by the 2-tape Büchi automaton $\mathcal{T}$ is the set of pairs $(u, v) \in \Sigma_1^\omega \times \Sigma_2^\omega$ such that there is some successful computation $\mathcal{C}$ of $\mathcal{T}$ over $(u, v)$.*

*The set of infinitary rational relations accepted by 2-tape Büchi automata will be denoted $RAT_2$.*

As noticed in the introduction an infinitary rational relation $R \subseteq \Sigma_1^\omega \times \Sigma_2^\omega$ may be considered as an $\omega$-language over the product alphabet $\Sigma_1 \times \Sigma_2$. We shall use this fact to investigate the topological complexity of infinitary rational relations.

## 3 Borel sets

We assume the reader to be familiar with basic notions of topology which may be found in [24, 19, 22, 30, 25].

For a finite alphabet $X$ we shall consider $X^\omega$ as a topological space with the Cantor topology. The open sets of $X^\omega$ are the sets in the form $W.X^\omega$, where $W \subseteq X^\star$. A set $L \subseteq X^\omega$ is a closed set iff its complement $X^\omega - L$ is an open set.

Define now the next classes of the Hierarchy of Borel sets of finite ranks:

**Definition 3.1** *The classes* $\mathbf{\Sigma^0_n}$ *and* $\mathbf{\Pi^0_n}$ *of the Borel Hierarchy on the topological space* $X^\omega$ *are defined as follows:*

$\mathbf{\Sigma^0_1}$ *is the class of open sets of* $X^\omega$.

$\mathbf{\Pi^0_1}$ *is the class of closed sets of* $X^\omega$.

*And for any integer* $n \geq 1$*:*

$\mathbf{\Sigma^0_{n+1}}$ *is the class of countable unions of* $\mathbf{\Pi^0_n}$*-subsets of* $X^\omega$.

$\mathbf{\Pi^0_{n+1}}$ *is the class of countable intersections of* $\mathbf{\Sigma^0_n}$*-subsets of* $X^\omega$.

The Borel Hierarchy is also defined for transfinite levels, but we shall not need them in the present study. There are also some subsets of $X^\omega$ which are not Borel. In particular the class of Borel subsets of $X^\omega$ is strictly included into the class $\mathbf{\Sigma^1_1}$ of analytic sets which are obtained by projection of Borel sets, see for example [30, 22, 25, 19] for more details.

Recall also the notion of completeness with regard to reduction by continuous functions. For an integer $n \geq 1$, a set $F \subseteq X^\omega$ is said to be a $\mathbf{\Sigma^0_n}$ (respectively, $\mathbf{\Pi^0_n}$, $\mathbf{\Sigma^1_1}$)-complete set iff for any set $E \subseteq Y^\omega$ (with $Y$ a finite alphabet): $E \in \mathbf{\Sigma^0_n}$ (respectively, $E \in \mathbf{\Pi^0_n}$, $E \in \mathbf{\Sigma^1_1}$) iff there exists a continuous function $f : Y^\omega \to X^\omega$ such that $E = f^{-1}(F)$.

A $\mathbf{\Sigma^0_n}$ (respectively, $\mathbf{\Pi^0_n}$, $\mathbf{\Sigma^1_1}$)-complete set is a $\mathbf{\Sigma^0_n}$ (respectively, $\mathbf{\Pi^0_n}$, $\mathbf{\Sigma^1_1}$)-set which is in some sense a set of the highest topological complexity among the $\mathbf{\Sigma^0_n}$ (respectively, $\mathbf{\Pi^0_n}$, $\mathbf{\Sigma^1_1}$)-sets. $\mathbf{\Sigma^0_n}$ (respectively, $\mathbf{\Pi^0_n}$)-complete sets, with $n$ an integer $\geq 1$, are thoroughly characterized in [29].

**Example 3.2** *Let $\Sigma = \{0,1\}$ and $\mathcal{A} = (0^\star.1)^\omega \subseteq \Sigma^\omega$. $\mathcal{A}$ is the set of $\omega$-words over the alphabet $\Sigma$ with infinitely many occurrences of the letter 1. It is well known that $\mathcal{A}$ is a $\mathbf{\Pi^0_2}$-complete set and its complement $\mathcal{A}^-$ is a $\mathbf{\Sigma^0_2}$-complete set: it is the set of $\omega$-words over $\{0,1\}$ having only a finite number of occurrences of letter 1.*

## 4 $\mathbf{\Pi^0_3}$-complete infinitary rational relations

We had got in [13] some $\mathbf{\Pi^0_3}$-complete infinitary rational relations. We used a coding of $\omega^2$-words over a finite alphabet $\Sigma$ by pairs of $\omega$-words over $\Sigma \cup \{A\}$ where $A$ is an additional letter not in $\Sigma$.

We shall modify the previous proof (only sketched in [13]) by coding an $\omega^2$-word over a finite alphabet $\Sigma$ by a **single** $\omega$-word over $\Sigma \cup \{A\}$. This way we can get some $\mathbf{\Pi^0_3}$-complete infinitary rational relation having some extra property.

**Theorem 4.1** *Let $\Gamma = \{0,1,A\}$ be an alphabet having three letters, and $\alpha$ be the $\omega$-word over the alphabet $\Gamma$ which is defined by:*

$$\alpha = A.0.A.0^2.A.0^3.A.0^4.A.0^5.A \ldots A.0^n.A.0^{n+1}.A \ldots \ .$$

*Then there exists an infinitary rational relation $R \subseteq \Gamma^\omega \times \Gamma^\omega$ such that:*

*$R_\alpha = \{\sigma \in \Gamma^\omega \mid (\sigma, \alpha) \in R\}$ is a $\mathbf{\Pi^0_3}$ − complete subset of $\Gamma^\omega$, and for all $u \in \Gamma^\omega - \{\alpha\}$ $\quad R_u = \{\sigma \in \Gamma^\omega \mid (\sigma, u) \in R\} = \Gamma^\omega$. Moreover $R$ is a $\mathbf{\Pi^0_3}$-complete subset of $\Gamma^\omega \times \Gamma^\omega$.*

**Proof.** We shall use a well known example of $\mathbf{\Pi^0_3}$-complete set which is a subset of the topological space $\Sigma^{\omega^2}$.

The set $\Sigma^{\omega^2}$ is the set of $\omega^2$-words over the finite alphabet $\Sigma$. It may also be viewed as the set of (infinite) $(\omega \times \omega)$-matrices whose coefficients are letters of $\Sigma$. If $x \in \Sigma^{\omega^2}$ we shall write $x = (x(m,n))_{m \geq 1, n \geq 1}$. The infinite word $x(m,1)x(m,2)\ldots x(m,n)\ldots$ will be called the $m^{th}$ column of the $\omega^2$-word $x$ and the infinite word $x(1,n)x(2,n)\ldots x(m,n)\ldots$ will be called the $n^{th}$ row of the $\omega^2$-word $x$. Thus an element of $\Sigma^{\omega^2}$ is completely determined by the (infinite) set of its columns or of its rows.

The set $\Sigma^{\omega^2}$ is usually equipped with the product topology of the discrete topology on $\Sigma$ (for which every subset of $\Sigma$ is an open set), see [19] [25]. This topology may be defined by the following distance $d$. Let $x$ and $y$ be two $\omega^2$-words in $\Sigma^{\omega^2}$ such that $x \neq y$, then

$$d(x,y) = \frac{1}{2^n} \ , \qquad \text{where}$$

$$n = min\{p \geq 1 \mid \exists (i,j)\ x(i,j) \neq y(i,j) \text{ and } i + j = p\} \ .$$

Then the topological space $\Sigma^{\omega^2}$ is homeomorphic to the above defined topological space $\Sigma^{\omega}$. The Borel hierarchy and the projective hierarchy on $\Sigma^{\omega^2}$ are defined from open sets in the same manner as in the case of the topological space $\Sigma^{\omega}$. The notion of $\mathbf{\Sigma^0_n}$ (respectively $\mathbf{\Pi^0_n}$)-complete sets is also defined in a similar way.

Let now

$$P = \{x \in \{0,1\}^{\omega^2} \mid \forall m \exists^{<\infty} n\ x(m,n) = 1\},$$

where $\exists^{<\infty}$ means "there exist only finitely many",

$P$ is the set of $\omega^2$-words having all their columns in the $\mathbf{\Sigma^0_2}$-complete subset $\mathcal{A}^-$ of $\{0,1\}^{\omega}$ where $\mathcal{A}$ is the $\mathbf{\Pi^0_2}$-complete $\omega$-regular language given in Example 3.2.

Recall the following classical result, [19, p. 179]:

**Lemma 4.2** *The set $P$ is a $\mathbf{\Pi^0_3}$-complete subset of $\{0,1\}^{\omega^2}$.*

**Proof.** Let $\mathcal{B}_m = \{x \in \Sigma^{\omega^2} \mid x(m,1)x(m,2)\dots x(m,n)\dots \in \mathcal{A}^-\}$ be the set of $\omega^2$-words over $\Sigma = \{0,1\}$ having their $m^{th}$ column in the $\mathbf{\Sigma_2^0}$-complete set $\mathcal{A}^-$. In order to prove that, for every integer $m \geq 1$, the set $\mathcal{B}_m$ is a $\mathbf{\Sigma_2^0}$-subset of $\Sigma^{\omega^2}$, consider the function $i_m : \Sigma^{\omega^2} \to \Sigma^\omega$ defined by $i_m(x) = x(m,1)x(m,2)\dots x(m,n)\dots$ for every $x \in \Sigma^{\omega^2}$. The function $i_m$ is continuous and $i_m^{-1}(\mathcal{A}^-) = \mathcal{B}_m$ holds. Therefore $\mathcal{B}_m$ is a $\mathbf{\Sigma_2^0}$-subset of $\Sigma^{\omega^2}$ because the class $\mathbf{\Sigma_2^0}$ is closed under inverse images by continuous functions.

Thus the set

$$P = \bigcap_{m \geq 1} \mathcal{B}_m$$

of $\omega^2$-words over $\Sigma$ having all their columns in $\mathcal{A}^-$ is a countable intersection of $\mathbf{\Sigma_2^0}$-sets so it is a $\mathbf{\Pi_3^0}$-set.

It remains to show that $P$ is $\mathbf{\Pi_3^0}$**-complete**. Let then $L$ be a $\mathbf{\Pi_3^0}$-subset of $\Sigma^\omega$. We know that $L = \cap_{i \in \mathbb{N}^*} A_i$ for some $\mathbf{\Sigma_2^0}$-subsets $A_i$, $i \geq 1$, of $\Sigma^\omega$. But $\mathcal{A}^-$ is $\mathbf{\Sigma_2^0}$-complete therefore, for each integer $i \geq 1$, there is some continuous function $f_i : \Sigma^\omega \to \Sigma^\omega$ such that $f_i^{-1}(\mathcal{A}^-) = A_i$.

Let now $f$ be the function from $\Sigma^\omega$ into $\Sigma^{\omega^2}$ which is defined by $f(x)(m,n) = f_m(x)(n)$. The function $f$ is continuous because each function $f_i$ is continuous.

For $x \in \Sigma^\omega$ $f(x) \in P$ iff the $\omega^2$-word $f(x)$ has all its columns in the $\omega$-language $\mathcal{A}^-$, i.e. iff for all integers $m \geq 1$

$$f_m(x) = f_m(x)(1)f_m(x)(2)\dots f_m(x)(n)\dots \in \mathcal{A}^-$$

iff $\forall m \geq 1 \quad x \in A_m$. Thus $f(x) \in P$ iff $x \in L = \cap_{m \geq 1} A_m$ so $L = f^{-1}(P)$.

We have then proved that all $\mathbf{\Pi_3^0}$-subsets of $\Sigma^\omega$ are inverse images by continuous functions of the $\mathbf{\Pi_3^0}$-set $P$ therefore $P$ is a $\mathbf{\Pi_3^0}$-complete set. □

In order to use this example we shall firstly define a coding of $\omega^2$-words over $\Sigma$ by $\omega$-words over the alphabet $(\Sigma \cup \{A\})$ where $A$ is a new letter not in $\Sigma$.

Let us call, for $x \in \Sigma^{\omega^2}$ and $p$ an integer $\geq 1$:

$$T^x_{p+1} = \{x(p,1), x(p-1,2), \ldots, x(2,p-1), x(1,p)\}$$

the set of elements $x(m,n)$ with $m+n = p+1$ and

$$U^x_{p+1} = x(p,1).x(p-1,2)\ldots x(2,p-1).x(1,p)$$

the sequence formed by the concatenation of elements $x(m,n)$ of $T^x_{p+1}$ for increasing values of $n$.

We shall code an $\omega^2$-word $x \in \Sigma^{\omega^2}$ by the $\omega$-word $h(x)$ defined by

$$h(x) = A.U^x_2.A.U^x_3.A.U^x_4.A.U^x_5.A.U^x_6.A\ldots A.U^x_n.A.U^x_{n+1}.A\ldots \ .$$

Let then $h$ be the mapping from $\Sigma^{\omega^2}$ into $(\Sigma \cup \{A\})^\omega$ such that, for every $\omega^2$-word $x$ over the alphabet $\Sigma$, $h(x)$ is the code of the $\omega^2$-word $x$ as defined above. It is easy to see, from the definition of $h$ and of the order of the enumeration of letters $x(m,n)$ in $h(x)$ (they are enumerated for increasing values of $m+n$), that $h$ is a continuous function from $\Sigma^{\omega^2}$ into $(\Sigma \cup \{A\})^\omega$.

Remark that the above coding of $\omega^2$-words resembles the use of the Cantor pairing function as it was used to construct the complete sets $P_i$ and $S_i$ in [31] (see also [29] or [30, section 3.4]).

**Lemma 4.3** *Let $\Sigma$ be a finite alphabet. If $L \subseteq \Sigma^{\omega^2}$ is $\mathbf{\Pi^0_3}$-complete then*

$$h(L) \cup h(\Sigma^{\omega^2})^-$$

*is a $\mathbf{\Pi^0_3}$-complete subset of $(\Sigma \cup \{A\})^\omega$.*

**Proof.** The topological space $\Sigma^{\omega^2}$ is compact thus its image by the continuous function $h$ is also a compact subset of the topological space $(\Sigma \cup \{A\})^\omega$. The set $h(\Sigma^{\omega^2})$ is compact hence it is a closed subset of $(\Sigma \cup \{A\})^\omega$ and its complement

$$(h(\Sigma^{\omega^2}))^- = (\Sigma \cup \{A\})^\omega - h(\Sigma^{\omega^2})$$

is an open (i.e. a $\mathbf{\Sigma_1^0}$) subset of $(\Sigma \cup \{A\})^\omega$.

On the other hand the function $h$ is also injective thus it is a bijection from $\Sigma^{\omega^2}$ onto $h(\Sigma^{\omega^2})$. But a continuous bijection between two compact sets is an homeomorphism therefore $h$ induces an homeomorphism between $\Sigma^{\omega^2}$ and $h(\Sigma^{\omega^2})$. By hypothesis $L$ is a $\mathbf{\Pi_3^0}$-subset of $\Sigma^{\omega^2}$ thus $h(L)$ is a $\mathbf{\Pi_3^0}$-subset of $h(\Sigma^{\omega^2})$ (where Borel sets of the topological space $h(\Sigma^{\omega^2})$ are defined from open sets as in the cases of the topological spaces $\Sigma^\omega$ or $\Sigma^{\omega^2}$).

The topological space $h(\Sigma^{\omega^2})$ is a topological subspace of $(\Sigma \cup \{A\})^\omega$ and its topology is induced by the topology on $(\Sigma \cup \{A\})^\omega$: open sets of $h(\Sigma^{\omega^2})$ are traces on $h(\Sigma^{\omega^2})$ of open sets of $(\Sigma \cup \{A\})^\omega$ and the same result holds for closed sets. Then one can easily show by induction that for every integer $n \geq 1$, $\mathbf{\Pi_n^0}$-subsets (resp. $\mathbf{\Sigma_n^0}$-subsets) of $h(\Sigma^{\omega^2})$ are traces on $h(\Sigma^{\omega^2})$ of $\mathbf{\Pi_n^0}$-subsets (resp. $\mathbf{\Sigma_n^0}$-subsets) of $(\Sigma \cup \{A\})^\omega$, i.e. are intersections with $h(\Sigma^{\omega^2})$ of $\mathbf{\Pi_n^0}$-subsets (resp. $\mathbf{\Sigma_n^0}$-subsets) of $(\Sigma \cup \{A\})^\omega$.

But $h(L)$ is a $\mathbf{\Pi_3^0}$-subset of $h(\Sigma^{\omega^2})$ hence there exists a $\mathbf{\Pi_3^0}$-subset $T$ of $(\Sigma \cup \{A\})^\omega$ such that $h(L) = T \cap h(\Sigma^{\omega^2})$. But $h(\Sigma^{\omega^2})$ is a closed i.e. $\mathbf{\Pi_1^0}$-subset (hence also a $\mathbf{\Pi_3^0}$-subset) of $(\Sigma \cup \{A\})^\omega$ and the class of $\mathbf{\Pi_3^0}$-subsets of $(\Sigma \cup \{A\})^\omega$ is closed under finite intersection thus $h(L)$ is a $\mathbf{\Pi_3^0}$-subset of $(\Sigma \cup \{A\})^\omega$.

Now $h(L) \cup (h(\Sigma^{\omega^2}))^-$ is the union of a $\mathbf{\Pi_3^0}$-subset and of a $\mathbf{\Sigma_1^0}$-subset of $(\Sigma \cup \{A\})^\omega$ therefore it is a $\mathbf{\Pi_3^0}$-subset of $(\Sigma \cup \{A\})^\omega$ because the class of $\mathbf{\Pi_3^0}$-subsets of $(\Sigma \cup \{A\})^\omega$ is closed under finite union.

In order to prove that $h(L) \cup (h(\Sigma^{\omega^2}))^-$ is **$\mathbf{\Pi_3^0}$-complete** it suffices to remark that

$$L = h^{-1}[h(L) \cup (h(\Sigma^{\omega^2}))^-] \ .$$

This implies that $h(L) \cup (h(\Sigma^{\omega^2}))^-$ is $\mathbf{\Pi_3^0}$-complete because $L$ is assumed to be $\mathbf{\Pi_3^0}$-complete. □

**Lemma 4.4** *Let* $P = \{x \in \{0,1\}^{\omega^2} \mid \forall m \exists^{<\infty} n \ x(m,n) = 1\}$ *and* $\Sigma = \{0,1\}$. *Then*

$$\mathcal{P} = h(P) \cup (h(\Sigma^{\omega^2}))^-$$

*is a* $\mathbf{\Pi_3^0}$*-complete subset of* $(\Sigma \cup \{A\})^\omega$.

**Proof.** It follows directly from the two preceding Lemmas. □

Let now $\Sigma = \{0,1\}$ and let $\alpha$ be the $\omega$-word over the alphabet $\Sigma \cup \{A\}$ which is defined by:

$$\alpha = A.0.A.0^2.A.0^3.A.0^4.A.0^5.A \ldots A.0^n.A.0^{n+1}.A \ldots \ .$$

We can now state the following Lemma.

**Lemma 4.5** *Let* $\Sigma = \{0,1\}$ *and* $\alpha$ *be the* $\omega$*-word over* $\Sigma \cup \{A\}$ *defined as above. Then there exists an infinitary rational relation* $R_1 \subseteq (\Sigma \cup \{A\})^\omega \times (\Sigma \cup \{A\})^\omega$ *such that:*

$$\forall x \in \Sigma^{\omega^2} \quad (x \in P) \text{ } \textit{iff} \text{ } ((h(x), \alpha) \in R_1) \ .$$

**Proof.** We define now the relation $R_1$. A pair $y = (y_1, y_2)$ of $\omega$-words over the alphabet $\Sigma \cup \{A\}$ is in $R_1$ if and only if it is in the form

$$y_1 = U_k.u_1.v_1.A.u_2.v_2.A.u_3.v_3.A \ldots A.u_n.v_n.A.\ldots$$

$$y_2 = V_k.w_1.z_1.A.w_2.z_2.A.w_3.z_3.A \ldots A.w_n.z_n.A \ldots$$

where $k$ is an integer $\geq 1$, $U_k, V_k \in (\Sigma^\star.A)^k$, and, for all integers $i \geq 1$,

$$v_i, w_i, z_i \in 0^\star \text{ and } u_i \in \Sigma^\star \text{ and}$$

$$|w_i| = |v_i| \quad \text{and} \quad [\ \ |u_{i+1}| = |z_i| + 1 \text{ or } |u_{i+1}| = |z_i| \ \ ]$$

and there exist infinitely many integers $i$ such that $|u_{i+1}| = |z_i|$.

We prove first that the relation $R_1$ satisfies:

$$\forall x \in \Sigma^{\omega^2} \quad (x \in P) \text{ iff } ((h(x), \alpha) \in R_1) \ .$$

Assume that for some $x \in \Sigma^{\omega^2}$ $\quad (h(x), \alpha) \in R_1$. Then $(h(x), \alpha)$ may be written in the above form $(y_1, y_2)$ with
$y_1 = U_k.u_1.v_1.A.u_2.v_2.A.u_3.v_3.A \ldots A.u_n.v_n.A.\ldots$
$y_2 = V_k.w_1.z_1.A.w_2.z_2.A.w_3.z_3.A \ldots A.w_n.z_n.A \ldots$

$y_1 = h(x)$ implies that for all integers $n \geq 1 \quad U^x_{k+n} = u_n.v_n$ thus $|u_n.v_n| = k + n - 1$.

$y_2 = \alpha$ implies that for all integers $n \geq 1 \quad w_n.z_n = 0^{k+n-1}$ thus $|w_n.z_n| = k + n - 1$.

So $|u_n.v_n| = |w_n.z_n|$ but by hypothesis $|w_n| = |v_n|$ therefore $|u_n| = |z_n|$.

Moreover $|u_{n+1}| = |z_n| + 1$ or $|u_{n+1}| = |z_n|$.

If $|u_{n+1}| = |z_n| + 1$ then $|u_{n+1}| = |u_n| + 1$ and $|v_{n+1}| = |v_n|$ because $|u_{n+1}| + |v_{n+1}| = |u_n| + |v_n| + 1$.

If $|u_{n+1}| = |z_n|$ then $|u_{n+1}| = |u_n|$ and $|v_{n+1}| = |v_n| + 1$ because $|u_{n+1}| + |v_{n+1}| = |u_n| + |v_n| + 1$.

This proves that the sequence $(|v_n|)_{n \geq 1}$ is increasing because for all integers $n \geq 1 \quad |v_{n+1}| = |v_n|$ or $|v_{n+1}| = |v_n| + 1$. Moreover by definition of $R_1$ we know that there exist infinitely many integers $n \geq 1$ such that $|u_{n+1}| = |z_n|$ hence also $|v_{n+1}| = |v_n| + 1$. Thus

$$\lim_{n \to +\infty} |v_n| = +\infty \ .$$

Let now $K$ be an integer $\geq 1$ and let us prove that the $K$ first columns of the $\omega^2$-word $x$ have only finitely many occurrences of the letter 1.

$\lim_{n \to +\infty} |v_n| = +\infty$ thus there exists an integer $N \geq 1$ such that $\forall n \geq N \quad |v_n| \geq K$.

Consider now, for $n \geq N$,

$$U^x_{k+n} = u_n.v_n = x(k+n-1,1).x(k+n-2,2)\ldots$$
$$\ldots x(2,k+n-2).x(1,k+n-1) \ .$$

We know that $v_n \in 0^\star$ thus

$$x(|v_n|, k+n-|v_n|) = x(|v_n|-1, k+n+1-|v_n|) = \ldots$$
$$\ldots = x(2,k+n-2) = x(1,k+n-1) = 0$$

and in particular

$$x(K, k+n-K) = x(K-1, k+n+1-K) = \ldots$$
$$\ldots = x(2, k+n-2) = x(1, k+n-1) = 0$$

because $|v_n| \geq K$.

These equalities hold for all integers $n \geq N$ and this proves that the $K$ first columns of the $\omega^2$-word $x$ have only finitely many occurrences of the letter 1.

But this is true for all integers $K \geq 1$ so **all columns** of $x$ have a finite number of occurrences of the letter 1 and $x \in P$.

Conversely it is easy to see that for each $x \in P$ the pair $(h(x), \alpha)$ may be written in the above form $(y_1, y_2) \in R_1$.

It remains only to prove that the above defined relation $R_1$ is an infinitary rational relation. It is easy to see that the following 2-tape Büchi automaton $\mathcal{T}$ accepts the infinitary rational relation $R_1$.

$\mathcal{T} = (K, \Gamma, \Gamma, \delta, q_0, F)$, where $K = \{q_0, q_1, q_2, q_3, q_4, q_5\}$ is a finite set of states, $\Gamma = \Sigma \cup \{A\} = \{0, 1, A\}$, with $\Sigma = \{0, 1\}$, $q_0$ is the initial state, and $F = \{q_4\}$ is the set of final states. Moreover $\delta \subseteq K \times \Gamma^\star \times \Gamma^\star \times K$ is the finite set of transitions, containing the following transitions:

$(q_0, a, \lambda, q_0)$, for all $a \in \Sigma$,
$(q_0, \lambda, a, q_0)$, for all $a \in \Sigma$,
$(q_0, A, A, q_0)$,
$(q_0, A, A, q_1)$,
$(q_1, a, \lambda, q_1)$, for all $a \in \Sigma$,
$(q_1, \lambda, \lambda, q_2)$,
$(q_2, a, 0, q_2)$, for all $a \in \Sigma$,
$(q_2, A, \lambda, q_3)$,
$(q_3, a, 0, q_3)$, for all $a \in \Sigma$,
$(q_3, \lambda, \lambda, q_4)$
$(q_3, a, \lambda, q_5)$, for all $a \in \Sigma$,
$(q_4, \lambda, A, q_2)$,
$(q_5, \lambda, A, q_2)$.

**Remark 4.6** *Using classical constructions from automata theory, we could have avoided the set of transitions to contain some transitions in the form* $(q_i, \lambda, \lambda, q_j)$*, like* $(q_1, \lambda, \lambda, q_2)$ *or* $(q_3, \lambda, \lambda, q_4)$.

**Lemma 4.7** *The set*

$$R_2 = (\Sigma \cup \{A\})^\omega \times (\Sigma \cup \{A\})^\omega - (h(\Sigma^{\omega^2}) \times \{\alpha\})$$

*is an infinitary rational relation.*

**Proof.** By definition of the mapping $h$, we know that a pair of $\omega$-words over the alphabet $(\Sigma \cup \{A\})$ is in $h(\Sigma^{\omega^2}) \times \{\alpha\}$ iff it is in the form $(\sigma_1, \sigma_2)$, where

$\sigma_1 = A.u_1.A.u_2.A.u_3.A.u_4.A \ldots .A.u_n.A.u_{n+1}.A \ldots$

$\sigma_2 = \alpha = A.0.A.0^2.A.0^3.A.0^4.A \ldots A.0^n.A.0^{n+1}.A \ldots$

where for all integers $i \geq 1$, $u_i \in \Sigma^\star$ and $|u_i| = i$.

So it is easy to see that $(\Sigma \cup \{A\})^\omega \times (\Sigma \cup \{A\})^\omega - (h(\Sigma^{\omega^2}) \times \{\alpha\})$ is the union of the sets $\mathcal{C}_j$ where:

- $\mathcal{C}_1 = \{(\sigma_1, \sigma_2) \ / \ \sigma_1, \sigma_2 \in (\Sigma \cup \{A\})^\omega \text{ and } (\sigma_1 \in \mathcal{B} \text{ or } \sigma_2 \in \mathcal{B})\}$

  where $\mathcal{B}$ is the set of $\omega$-words over $(\Sigma \cup \{A\})$ having only a finite number of letters $A$.

- $\mathcal{C}_2$ is formed by pairs $(\sigma_1, \sigma_2)$ where

  $\sigma_1$ or $\sigma_2$ has not any initial segment in $A.\Sigma.A.\Sigma^2.A$.

- $\mathcal{C}_3$ is formed by pairs $(\sigma_1, \sigma_2)$ where

  $\sigma_2 \notin \{0, A\}^\omega$.

- $\mathcal{C}_4$ is formed by pairs $(\sigma_1, \sigma_2)$ where

  $\sigma_1 = A.w_1.A.w_2.A.w_3.A.w_4.A \ldots A.w_n.A.u.A.z_1$

  $\sigma_2 = A.w'_1.A.w'_2.A.w'_3.A.w'_4.A \ldots A.w'_n.A.v.A.z_2$

  where $n$ is an integer $\geq 1$, for all $i \leq n \quad w_i, w'_i \in \Sigma^\star$, $z_1, z_2 \in (\Sigma \cup \{A\})^\omega$ and

$$u, v \in \Sigma^\star \text{ and } |v| \neq |u|$$

- $\mathcal{C}_5$ is formed by pairs $(\sigma_1, \sigma_2)$ where

  $\sigma_1 = A.w_1.A.w_2.A.w_3.A.w_4 \ldots A.w_n.A.w_{n+1}.A.v.A.z_1$

  $\sigma_2 = A.w'_1.A.w'_2.A.w'_3.A.w'_4 \ldots A.w'_n.A.u.A.z_2$

  where $n$ is an integer $\geq 1$, for all $i \leq n$ $w_i, w'_i \in \Sigma^\star$, $w_{n+1} \in \Sigma^\star$, $z_1, z_2 \in (\Sigma \cup \{A\})^\omega$ and

$$u, v \in \Sigma^\star \text{ and } |v| \neq |u| + 1 \ .$$

Each set $\mathcal{C}_j$, $1 \leq j \leq 5$, is easily seen to be an infinitary rational relation $\subseteq (\Sigma \cup \{A\})^\omega \times (\Sigma \cup \{A\})^\omega$ (the detailed proof is left to the reader). The class $RAT_2$ is closed under finite union thus

$$R_2 = (\Sigma \cup \{A\})^\omega \times (\Sigma \cup \{A\})^\omega - (h(\Sigma^{\omega^2}) \times \{\alpha\}) = \bigcup_{1 \leq j \leq 5} \mathcal{C}_j$$

is an infinitary rational relation. □

Return now to the proof of Theorem 4.1. Let

$$R = R_1 \cup R_2 \subseteq \Gamma^\omega \times \Gamma^\omega.$$

The class $RAT_2$ is closed under finite union therefore $R$ is an infinitary rational relation.

Lemma 4.5 and the definition of $R_2$ imply that $R_\alpha = \{\sigma \in \Gamma^\omega \mid (\sigma, \alpha) \in R\}$ is equal to the set $\mathcal{P} = h(P) \cup (h(\Sigma^{\omega^2}))^-$ which is a $\mathbf{\Pi^0_3}$-complete subset of $(\Sigma \cup \{A\})^\omega$ by Lemma 4.4.

Moreover, for all $u \in \Gamma^\omega - \{\alpha\}$, $R_u = \{\sigma \in \Gamma^\omega \mid (\sigma, u) \in R\} = \Gamma^\omega$ holds by definition of $R_2$.

In order to prove that $R$ is a $\mathbf{\Pi^0_3}$-set remark first that $R$ may be written as the union:

$$R = \mathcal{P} \times \{\alpha\} \bigcup \Gamma^\omega \times (\Gamma^\omega - \{\alpha\}).$$

We already know that $\mathcal{P}$ is a $\mathbf{\Pi^0_3}$-complete subset of $(\Sigma \cup \{A\})^\omega$. Then it is easy to show that $\mathcal{P} \times \{\alpha\}$ is also a $\mathbf{\Pi^0_3}$-subset of $(\Sigma \cup \{A\})^\omega \times (\Sigma \cup \{A\})^\omega$. On the other side it is easy to see that $\Gamma^\omega \times (\Gamma^\omega - \{\alpha\})$ is

an open subset of $\Gamma^\omega \times \Gamma^\omega$. Thus $R$ is a $\mathbf{\Pi^0_3}$-set because the Borel class $\mathbf{\Pi^0_3}$ is closed under finite union.

Moreover let $g : \Sigma^{\omega^2} \to (\Sigma \cup \{A\})^\omega \times (\Sigma \cup \{A\})^\omega$ be the function defined by:

$$\forall x \in \Sigma^{\omega^2} \qquad g(x) = (h(x), \alpha).$$

It is easy to see that $g$ is continuous because $h$ is continuous. By construction it turns out that for all $\omega^2$-words $x \in \Sigma^{\omega^2}$ $\quad (x \in P)$ iff $(g(x) \in R)$. This means that $g^{-1}(R) = P$. This implies that $R$ is $\mathbf{\Pi^0_3}$-complete because $P$ is $\mathbf{\Pi^0_3}$-complete. □

**Remark 4.8** *The structure of the $\mathbf{\Pi^0_3}$-complete infinitary rational relation $R$ we have just got is very different from the structure of a previous example given in [13]. It can be described very simply by the sections $R_u$, $u \in \Gamma^\omega$. All sections but one are equal to $\Gamma^\omega$, so they have the lowest topological complexity and exactly one section is a $\mathbf{\Pi^0_3}$-complete subset of $\Gamma^\omega$.*

**Acknowledgements.** Thanks to Jean-Pierre Ressayre and Pierre Simonnet for useful discussions.

## References

[1] Ya M. Barzdin and B.A. Trakhtenbrot. *Finite Automata, Behaviour and Synthesis*, Nauka, Moscow, 1970 (English translation, North Holland, Amsterdam, 1973).

[2] M.-P. Béal and O. Carton. *Determinization of Transducers over Infinite Words*, in ICALP'2000 (U. Montanari et al., eds.), vol. 1853 of Lect. Notes in Comput. Sci., pp. 561–570, 2000.

[3] M.-P. Béal, O. Carton, C. Prieur and J. Sakarovitch. *Squaring Transducers: An Efficient Procedure for Deciding Functionality and Sequentiality*, Theoretical Computer Science, vol. 292, no. 1, pp. 45–63, 2003.

[4] J. Berstel. *Transductions and Context Free Languages*, Teubner Verlag, 1979.

[5] J.R. Büchi. *On a Decision Method in Restricted Second Order Arithmetic, Logic Methodology and Philosophy of Science*, (Proc. 1960 Int. Congr.), Stanford University Press, 1962, 1–11.

[6] T. Cachat, J. Duparc and W. Thomas. *Solving Pushdown Games with a $\Sigma_3$ Winning Condition*, proceedings of CSL 2002, Lecture Notes in Computer Science, Springer, Volume 2471, pp. 322–336,

[7] C. Choffrut. *Une Caractérisation des Fonctions Séquentielles et des Fonctions Sous-Séquentielles en tant que Relations Rationnelles*, Theoretical Computer Science, Volume 5, 1977, pp.325–338.

[8] C. Choffrut and S. Grigorieff. *Uniformization of Rational Relations*, Jewels are Forever 1999, J. Karhumäki, H. Maurer, G. Paun and G. Rozenberg editors, Springer, pp.59–71.

[9] J. Engelfriet and H. J. Hoogeboom. *X-automata on $\omega$-Words*, Theoretical Computer Science, Volume 110, (1993) 1, 1-51.

[10] O. Finkel. *Topological Properties of Omega Context Free Languages*, Theoretical Computer Science, Volume 262 (1-2), 2001, pp.669–697.

[11] O. Finkel. *On the Topological Complexity of Infinitary Rational Relations*, RAIRO-Theoretical Informatics and Applications, Volume 37 (2), 2003, pp. 105–113.

[12] O. Finkel. *Undecidability of Topological and Arithmetical Properties of Infinitary Rational Relations*, RAIRO-Theoretical Informatics and Applications, Volume 37 (2), 2003, pp.115–126.

[13] O. Finkel. *On Infinitary Rational Relations and Borel Sets*, in the Proceedings of the Fourth International Conference on Discrete Mathematics and Theoretical Computer Science DMTCS'03, 7 - 12 July 2003, Dijon, France, Lecture Notes in Computer Science, Springer, Volume 2731, pp. 155–167.

[14] O. Finkel. *Borel Ranks and Wadge Degrees of Omega Context Free Languages*, in the Proceedings of New Computational Paradigms: First Conference on Computability in Europe, CiE 2005, Amsterdam, The Netherlands, Lecture Notes in Computer Science, Volume 3526, Springer, 2005, pp.129–138.

[15] O. Finkel. *On the Accepting Power of 2-Tape Büchi Automata*, in the Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science, STACS 2006, Marseille, France, February 23-25, 2006, Lecture Notes in Computer Science, Volume 3884, Springer, Volume 3884, pp.301–312.

[16] F. Gire. *Relations Rationnelles Infinitaires*, Thèse de troisième cycle, Université Paris 7, Septembre 1981.

[17] F. Gire. *Une Extension aux Mots Infinis de la Notion de Transduction Rationnelle*, 6th GI Conf., Lecture Notes in Computer Science, Volume 145, 1983, pp.123–139.

[18] F. Gire and M. Nivat. *Relations Rationnelles Infinitaires*, Calcolo, Volume XXI, 1984, pp.91–125.

[19] A.S. Kechris. *Classical Descriptive Set Theory*, Springer-Verlag, 1995.

[20] K. Kuratowski. *Topology*, Academic Press, New York 1966.

[21] L. H. Landweber. *Decision Problems for $\omega$-Automata*, Math. Syst. Theory 3 (1969) 4, 376–384.

[22] H. Lescow and W. Thomas. *Logical Specifications of Infinite Computations*, In:"A Decade of Concurrency" (J. W. de Bakker et al., eds), Lecture Notes in Computer Science, Springer, Volume 803 (1994), 583–621.

[23] R. Lindner and L. Staiger. *Algebraische Codierungstheorie - Theorie der Sequentiellen Codierungen*, Akademie-Verlag, Berlin, 1977.

[24] Y. N. Moschovakis. *Descriptive Set Theory*, North-Holland, Amsterdam 1980.

[25] D. Perrin and J.-E. Pin. *Infinite Words, Automata, Semigroups, Logic and Games*, Volume 141 of Pure and Applied Mathematics, Elsevier, 2004.

[26] J-E. Pin. *Logic, Semigroups and Automata on Words*, Annals of Mathematics and Artificial Intelligence 16 (1996), pp.343–384.

[27] C. Prieur. *Fonctions Rationnelles de Mots Infinis et Continuité*, Thèse de Doctorat, Université Paris 7, Octobre 2000.

[28] P. Simonnet. *Automates et Théorie Descriptive*, Thèse de Doctorat, Université Paris 7, March 1992.

[29] L. Staiger. *Hierarchies of Recursive $\omega$-Languages*, Jour. Inform. Process. Cybernetics EIK 22 (1986) 5/6, 219–241.

[30] L. Staiger. *$\omega$-Languages*, Chapter of the Handbook of Formal languages, Vol 3, edited by G. Rozenberg and A. Salomaa, Springer-Verlag, Berlin.

[31] L. Staiger and K. Wagner. *Rekursive Folgenmengen I*, Z. Math Logik Grundlag. Math. 24, 1978, 523–538.

[32] W. Thomas. *Automata and Quantifier Hierarchies*, in: Formal Properties of Finite automata and Applications, Ramatuelle, 1988, Lecture Notes in Computer Science 386, Springer, Berlin, 1989, pp.104–119.

[33] W. Thomas. *Automata on Infinite Objects*, in: J. Van Leeuwen, ed., Handbook of Theoretical Computer Science, Vol. B (Elsevier, Amsterdam, 1990), pp. 133–191.

Olivier Finkel,

Equipe Modèles de Calcul et Complexité
Laboratoire de l'Informatique du Parallélisme
CNRS et Ecole Normale Supérieure de Lyon
46, Allée d'Italie 69364 Lyon Cedex 07, France.
E–mail: *Olivier.Finkel@ens − lyon.fr*

Received February 1, 2007

Computer Science Journal of Moldova, vol.**15**, no.1(43), 2007

# Machine Intelligence Quotient as a Complex Fuzzy Numeral

V. C. I. Ulinwa

**Abstract**

**Abstract.** An ongoing research shows that machine intelligence quotient (MIQ) is an integrated complex numeral from three standard measures and transformable within the plane and other coordinates. With distinctive scales, technical, personal, and legislative, the multiple perspectives inquiring system (TOP) is used in calibrating, measuring, and interpreting the quotient. Given the homogeny of the linguistic Choquet fuzzy integral and linguistic complex fuzzy set theorems, on which the considered machine intelligence measurement is based, a new MIQ calculus is presented for consideration. The tenets are expected to withstand technological advancement and human interpretation.

**Keywords**. Machine Intelligence Measurement, MIQ, Machine Intelligence Quotient, Multiple Perspective Inquiring Analysis, TOP

## 1 Introduction

The investigated phenomenon, machine intelligence quotient, is controversial and important for electromechanical advancement [29]. One of the controversies is how to determine machines that think [28]. The other is how to figure their level of intelligence and representation [7] [13] [24] [30]. According to [12], machine intelligence is a result of some type of rules that are algorithmically coded on software or hardwired.

Although Zadeh [32] envisaged the concept of machine intelligence quotient (MIQ), Turing [28] testing primer and Searle [24] argument

advanced the necessity as much as Descente's work laid the fundamental need [29] [31]. Ironically, little is still known about MIQ [3]. For these reasons, this study used the multiple perspective inquiring method (TOP) to elucidate a new measurement method. The invocation of ambiguous quantified tenets similar to human intelligence measurement is avoided. Rather MIQ is factually an aggregation of disjointed complex fuzzy sets, a nonunitary definition. It is unwise to use a single and the same indicator to qualitatively or quantitatively represent the intelligence or to use the indices of quality to presume quantity and visa-vis. This investigation also takes exception in equating performance as if it is the intelligence. Such a supposition is far fetched; performance measure is a sub measure of productivity. Rather, machine intelligence is relative to productivity because machine intelligence without a productive work is a waste. Also, human preference for quality than quantity and quantitative desire for more of quality things, with respect to machine intelligence, is reconciled. The quality is not about the outward appearance of a machine but about the implicit tangible. As such, this study uses a Cartesian fuzzy set to represent the intelligence.

## 2 Multiple Perspective Inquiring Method

To provide a valid and reliable measurement instrument that captures features in diverse but correlated machine intelligence domains, TOP scales are used. TOP minimizes statistical biases that are common in the quantitative science and practice; secondly, it discovers the underlying perspective meanings that affect the science of machine intelligence; three, it ratifies theory and data anchors of the intelligence that ground on more than one perspective; and four, it insures that the bases of any solution and thesis are within the domain peer experts and consumers recognize.

The presupposed calibration approach is grounded on the derivatives set forth by [16], [17], [22], [26], and [33] because it is imperative to lay down a comprehensive and standard method for measuring the intelligence. Moreover, TOP brings to bear, in any given machine in-

telligence measurement the factors [16].

## 2.1 Technical Perspective (T)

The quantitative science of machine intelligence measurement requires T perspective, a tenet that numerically justifies every means and results. It uses the science to isolate, abstract, idealize, and simplify problems into solutions [22]. For the measurement to be a major scientific function, results must be quantitatively analyzed, interpreted, and reported.

A five-theoretic topology [6], with a distinct name of a philosopher: Leibniz, Locke, Kant, Hegel, and Singer, is crucially used to explain this perspective [8], [16]. For the Leibnizian, truth is analytical and can be mathematically reduced into a solution space. To the Lockean, truth is experimental and in any given problem peer experts' scientific opinion determines if a solution is acceptable or not. The Kantian inquiring analysis rests on the assumption that truth is synthetic and only through two complementary solution models. Null and alternative hypotheses are developed for accepting or rejecting any practice that is hard to be studied with the Lockean or the Leibnzian method. To the Hegelian, analysis is grounded on the premise that truth conflicts and only through formulation of antithetical representation. The Singerian inquiring analysis emphasizes on pragmatic methods relative to the general purpose and objective of an inquiry [6],[16].

## 2.2 Organizational Perspective (O)

Organizational filter, legislative filter as it is in this case, is for observing and analyzing an organization's tenets of machine intelligence. The O perspective relies on policies and ethics. For example, it insures that the intelligence is within the acceptable scientific practices. It determines the standard and conditions for rigorous issues. Generally, the O perspective does not seek optimal solutions but emphasizes on compromise and routine.

### 2.3 Personal Perspective (P)

The personal perspective is very subtle compared to the others. It brings to bear the psychology, ethics, and sociology of those whose decisions affect machine intelligence, and these factors are inseparable from any model [8], [16]. It brings human persona or the eye of an individual into measurement science and practice. It is the unique insight and intuition for analysis [16].

## 3 Machine Intelligence Measurement

When one observes certain machinery systems, there abound manifests that appeal and in some cases are the cursors of the intelligence. As such, a scientific approach is needed to determine and measure the intelligence in controllable scientific contexts.

### 3.1 Contexts

The first thing is to define the appropriate contexts. The contexts are the informational descriptions and explanations about the system, its domain, resources available to it, and contribution to humanity. They characterize the situations. Events are just the sub-summaries of the contexts.

But in what contexts should the intelligence be tested? One should avoid testing only in contexts in which the purpose of the intelligence is predictable or obvious. This approach is not promising for the system behavior is known. The test, instead, should be conducted in controlled contexts such as normal, sudden, rare, and where or when the purpose is barely present.

Events in a normal context are the types system designers generally model for. They are well defined and predictable and usually unwanted events are generally minimized or eliminated; sudden events occur unexpectedly; rare events are those the designers consider to be possible but least probable and they are sometimes quite catastrophic or beneficial and very difficult to detect; and less purposeful event is one in which the purpose of the system is barely present in the environment.

Given the contextual event, the focus should be on what Ji and Chen [14] characterized as knowledge incompleteness, motivation correlativeness, and initiative openness. Knowledge incompleteness occurs when events are so-new to a system and it is unable to deduce knowledge from its base. On the other hand, motivation correlativeness is due to events that positively or negatively impact the achievement of goals. Finally, initiative openness helps explain opportunities that partially open to a system control for promoting benefits or reducing and avoiding cost relative to measurable. Numbered equations must be managed manually.

### 3.2 Measurable

Although there is no general list of acceptable factors of machine intelligence, the identified relevant qualitative and quantitative measurable or factors, used in this study, are impedance, machine (process) capability, productivity, versatility, and agility, to list a few. Figure 1 shows how conventional measures, with respect to the measurable, are sorted as T, O, or P.

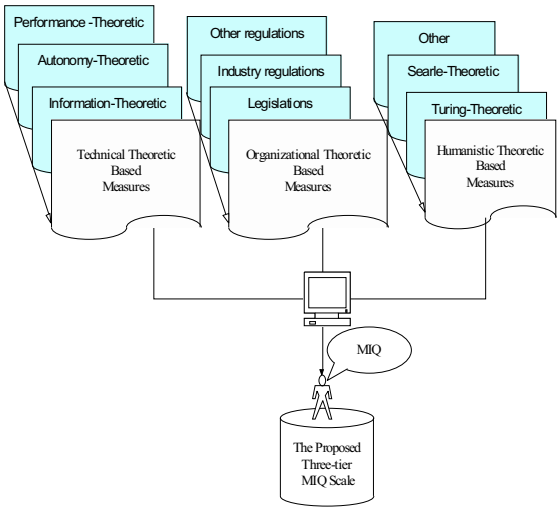


Figure 1. TOP Classified Measurable

From Figure 1 it is evident that Turing test and the Searle's argument are within the humanistic category and the information-theoretic or autonomous theoretic measures are in the technical kind. To the technical school, machine intelligence is all but qualitative manifest. This induces the search for a universal numerical meaning such that any qualitative feature is unscientific. The purpose is to operationalize properties of the intelligence in terms of behavioral actions that can be mathematically measured and manipulated.

Measuring machine intelligence is elusive and subtle with the personal perspective than the T or O. The qualitative measure is usually from an individual's eyes and mental representation; and it is grounded on human charisma and interest. From this point of view, it is the persona of human intelligence and it filters in qualities parallel to the manifests of human neural and social implications. In other words, it is a general logic that contains no precepts but a rule-governed manifestation [15]. It is therefore evident from Figure 1 that measurable for T oriented measures such as information or performance measures are different from those for O and P. The standardization consists of using T to quantify factors such as productivity, O to assess compliance of the intelligence to the relevant regulations, and P to assess the socio-psychological aspects. The latter should include the concerns of Turing [28] and Searle [23]; and it is like tasting wine or rating movies or music.

To start with, let $O$ be a set of observations such that $o_i \in O$ and $b_i \in B$ represent a time series observable machine behaviors during an event type $e_i$. If $e_i \in E$, $b_i \in B$, and $o_i \in O$ then the following are:

1. Determining relevant measurable relative to machine intelligence;
2. Determining machine productivity during each event type;
3. Deriving a technical measure of machine intelligence;
4. Deriving a legislative measure of machine intelligence;
5. Deriving a humanistic measure of machine intelligence; and

## 6. Determining MIQ from 3, 4, and 5 relative to the effects of 1 and 2

Given these conditions, the first thing is to convert T, O, P measures to complex fuzzy sets. The procedural method starts with fuzzy linguistic variables [18] [19]. A linguistic variable is a quintuple $(x, T\,(x)\,U, G, M)$, where $x$ is a variable, *T(x)* is a set of the variables in a universe $U$ and $G$ is a syntactic rule that generates the linguistic values. M is a rule relating meanings of the linguistic values such that fuzzy relations, the interaction between the components of complex fuzzy numbers, are meaningful [20] [21].

Thus T, O, and P as linguistic variables consist of laced labels; each label specifies a focal point of the measurement. The most common methods are a fuzzy triangular, a trapezoidal, and an exponential functions as shown in Equations (1), (2), and (3) where LS means linguistic set [25]; and resemble the ones shown in Figure 3:

$$L_s = \int_{-2}^{0} \left(\tfrac{2+x}{2}\right) \Big/ x + \int_{0}^{2} \left(\tfrac{2-x}{2}\right) \Big/ x \tag{1}$$

$$L_s = \int_{-4}^{2} \left(\tfrac{4+x}{2}\right) \Big/ x + \int_{2}^{2} 1/x + \int_{2}^{4} \left(\tfrac{4-x}{2}\right) \Big/ x \tag{2}$$

$$L_s = \int_{x} e^{-5(x-5)^2}/x \tag{3}$$

In Figure 2, (a) is a triangular set, (b) is a trapezoidal set, and (c) is a Gaussian set. Notice that the sets are the result of a membership function that $f(A) \longrightarrow [0, 1]$ assigns numbers in $[0, 1]$ interval to subsets of a universe of discourse $\mu$ [9] [10] [18]. Using scale 5.45c, number 3, from [4] each of the labels of T, O, and P measures is characterized as 'very unsatisfactory', 'unsatisfactory', 'partially satisfactory', 'satisfactory', or 'very satisfactory' with unique fuzzy numbers [4] [5]. With Equations (4) and (5), fuzzy projection, each of the linguistic variables is transformed to a linguistic complex fuzzy set such that [19]:

$$proj[R; X] = \int_{x} \left(\max_{y} \mu_R(x, y)\right)/x \tag{4}$$

$$proj[R; Y] = \int_{x} \left(\max_{x} \mu_R(x, y)\right)/y \tag{5}$$

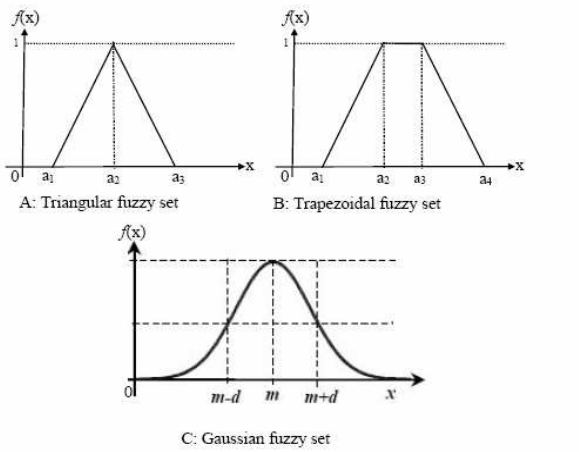


Figure 2. Common Membership Types

Although [10] suggested $\overline{X} = \int_{Ux} [\subset U_y \wedge \mu_R(x,y)]\,/x$ and $\overline{Y} = \int_{Uy} [\subset U_x \wedge \mu_R(x,y)]\,/y$ for completing the transformation, linguistic Choquet fuzzy integral [1] [2] [10] is recommended instead. The resulting set is the linguistic complex fuzzy set [2].

# 4 Complex Fuzzy Set

## 4.1 Background

A linguistic complex fuzzy set $z = x + jy$, in a Cartesian coordinate, is composed of two linguistic fuzzy sets [19]. One for a real number Re(z) and the other for the imaginary part Im(z). Each relates to a specific dimension such that $z = x + jy$ is information on the X-axis and the Y-axis of the set. Intuitively, arithmetic operations can be performed on such sets [9] [19] [20] [21]. Given two such sets $Z_1 = (x_1 + jy_1)$ and $Z_2 = (x_2 + jy_2)$ then $Z = Z_1 + Z_2 = (x_1 + x_2) + (jy_1 + jy_2)$. Because [9] noted that it is easy to perform division operation in polar plane,

$Z$ is transformed to a polar plane such that $p^{\theta} = Z = (x + jy)$ where $\mu_1 (x + jy) = \theta$ and $\theta = p = \sqrt{x^2 + y^2}$. Notice that $\theta = \arctan\left(\frac{y}{x}\right)$. If $Z_1 Z_2$ or $\frac{Z_1}{Z_2}$ is in a polar plane then $Z_1$ and $Z_2$are also polar coordinated [19]. These operations are applicable to machine intelligence measurement.

## 4.2 Application to Machine Intelligence Measurement

Like any other implicit tangible goods, machine intelligence is a purposeful creation. Setting it relative to economic values is intuitive and meaningful for productive endeavors. Therefore, the first step is to set machine intelligence relative to $O$ or a compliance measure. Figure 3 is an example. This is done by fixing compliance along the $X$ axis as $x$ for $T$ or $P$ along the $Y$ axis. The fixation, an aggregated value, is determined with linguistic Choquet fuzzy integral [1] [2].

With $(C)\int f d\mu = \sum_{i=1}^{n} (h_i - r_{i-1})\mu(A_i)$, a standard Choquet fuzzy integral defined over nonadditive measures [10],

$$\int_c h \circ g = \sum_{i=1}^{n} h(x_i)\left[g(X_i) - g(x_{i-1})\right] \tag{6a}$$

or

$$\int_c h \circ g = \sum_{i=1}^{n} g(x_i)\left[h(X_i) - h(x_{i-1})\right] \tag{6b}$$

defines the lower and upper bounds of the linguistic fuzzy sets intervals such that

$$\int_c h \circ g = \bigcup_{\alpha \in [0,1]} \left[\int_c h \circ g\right]. \tag{6c}$$

It is evident from [1] [2] that such

$$\left[\int_c h \circ g\right] = \left[\int_c \underline{[h]}_{\alpha} \circ g, \overline{[h]}_{\alpha} \circ g\right] \tag{6d}$$

is a linguistic Choquet fuzzy integral where

$\underline{[h]}_{\alpha} = [h]_{\alpha}$, $\overline{[h]}_{\alpha}$, $\left\{h \in R | \underline{[h]}_{\alpha} \leq \overline{[h]}_{\alpha}\right\}$, and $0 \leq \alpha \leq 1$.

Given the integration, it is clear that subsethood of $T$ and $P$ quantifies MIQ after conditioning $T$ and $P$ to the compliance measure $O$; meaning that

$$MIQ = S(T,P) = \left(\frac{|T \cap P|}{|T|}\right)\left(\frac{|T \cap P|}{|P|}\right) \tag{7}$$

with $\cap$ as a standard fuzzy interception operator.

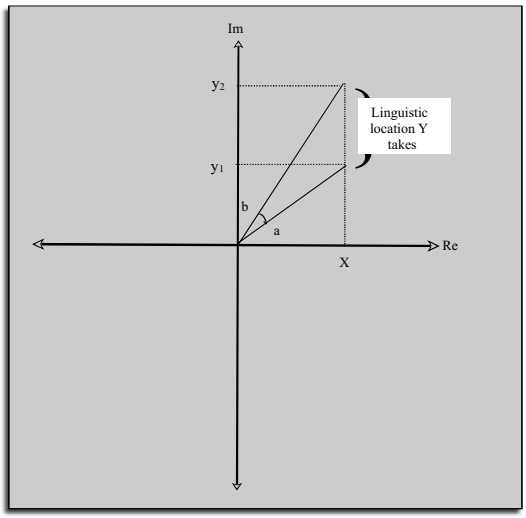


Figure 3. Common Membership Types

To profile the machine given a contextual event and taking linguistic variables or fuzzy numbers as inputs, $m^T$ observations are taken on $n$ measurable. If $t_{ij} \subset T$, $l_{ij} \subset O$, and $p_{ij} \subset O$ measure the system behavior such that $f(x_{ij}) = [t_{ij}, l_{ij}, p_{ij}]$ then

$$T_{ij} = \frac{t_{ij}}{\sum_{i,j=1} t_{ij}},\ L_{ij} = \frac{l_{ij}}{\sum_{i,j=1} l_{ij}},\ \text{and}\ P_{ij} = \frac{p_{ij}}{\sum_{i,j=1} p_{ij}}. \tag{8}$$

Each is further normalized with its direct variance with golden ratio such that $y = kw$, where $k$ is the constant of proportionality of the greatest $j^{th}$ column value and golden ratio: 1.618. Notice that $y$ varies as $x$; and $k$ is used to obtain new normalized $T_{ij}, L_{ij}, P_{ij}$ values.

Next, a composition of fuzzy relation on each event $E_i$ and its transpose $E^{-1}$ is taken such that

$$E \circ E^{-1} = \begin{bmatrix} r_{11} & \cdots & r_{1m} \\ : & : & : \\ r_{n1} & \cdots & r_{mn} \end{bmatrix} \circ \begin{bmatrix} r_{11} & \cdots & r_{n1} \\ : & : & : \\ r_{1m} & \cdots & r_{mn} \end{bmatrix} \tag{9}$$

where $E \circ E^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \circ \begin{bmatrix} a & c \\ b & d \end{bmatrix} =$

$$= \begin{bmatrix} a \cap a & \cup & b \cap b & & a \cap c & \cup & b \cap d \\ c \cap a & \cup & d \cap b & & c \cap c & \cup & d \cap d \end{bmatrix}.$$

It involves taking the maximum of the minimums per row of a fuzzy projection. Finally, from [5] a matched ratio (MR) of the projected fuzzy set and each of the reference set is calculated using

$$MR = \frac{1}{2} \frac{\int_{s_y} \mu_{\overline{y}}(y)dy + \int_{s_y} \mu_{\widetilde{\overline{y}}}(y)dy - D}{\int_{s_y} \mu_{\overline{y}}(y)dy}, \tag{10}$$

where $D = \int s_{\overline{y}} \cup s_{\widetilde{\overline{y}}} | \mu_{\overline{y}}(y) - \mu_{\widetilde{\overline{y}}} | dy$ .

Given a highest matched ratio, the idea is to select the reference set corresponding with the projected set. The selected set then becomes the linguistic variable representing the profile. Alternatively, the crispy numbers that represent the reference sets could be used directly for the measurement. Letting $\widetilde{L}_{ij}$ be the linguistic variables, $N = \left\{ \widetilde{L}_{11}, \widetilde{L}_{12} ... \widetilde{L}_{1m} \right\}$, and knowing that $\widetilde{L}_i = [a_i, b_i, c_i, d_i]$ and $i \in 1, 2, ..., n$, trapezoid with parameters defined by

$$\overline{\widetilde{L}}_i = \tfrac{1}{n} \bigoplus_{j=1}^{m} \widetilde{L}_{ij}, \tag{11}$$

then Equation (11) is the extension principle of addition of fuzzy means [4]. The steps are repeated for all the measurement contexts. Thus with a given $n$ number of measurable there are associable $n$ rated machine behavior and $n$ productivity to explain the system's profile from a linguistic variable.

# 5 Conclusion

When one observes a machinery system there abound manifests that appeal and in some cases act as cursors for attributing intelligence to it. What is arguable is how to measure the intelligence ordained on them. Equally controversial is how to represent them. The postulation used in this paper consists of three perspectives: technical (T), organizational (O), and personal (P). Each calibrates the intelligence differently. The T is the most traditional, quantitative, method of measuring observation; the O measures compliance of a machine to certain legislated criteria; and the P takes humanities into the measurement and interpretation.

Because each gives a different definition and measure of the intelligence, it is only when T, O, and P are integrated that one derives a meaningful measure. This is important when a machinery system manufacturer, regulatory agency, or a user ascribes intelligence to the orderliness; or an observer tests if a machine is intelligent. The endeavor is that a machinery system is not intelligent until it is measured against the required purpose, compliance to a certain regulation, and what humans understand as showing intelligence.

For each defined measurement context, the T filter should be used to quantify the manifest using input and output relations. Focus should also be on machine productivity relative to each context. The O perspective should be used to measure compliance of the system to any regulation during each context. This requires using any standard compliance measure as assessment tool or a new one constructed. This measure is important because any intelligent system that is not incompliance, for instance, to a required act of congress or an executive order is stupid with respect to this perspective. In addition to the T and O calibrations per the contexts, P requires the use of socio-psychology scale.

Figure 4 shows the independent filters as used for defining and measuring the intelligence. The presupposed concern is the region of interception. The region is where all agree on what is intelligent. The sets can overlap more or be null depending on the system behavior. So

far the indices are meaningless until cross-cued. The meaningfulness necessitates linguistic complex fuzzy set theory.

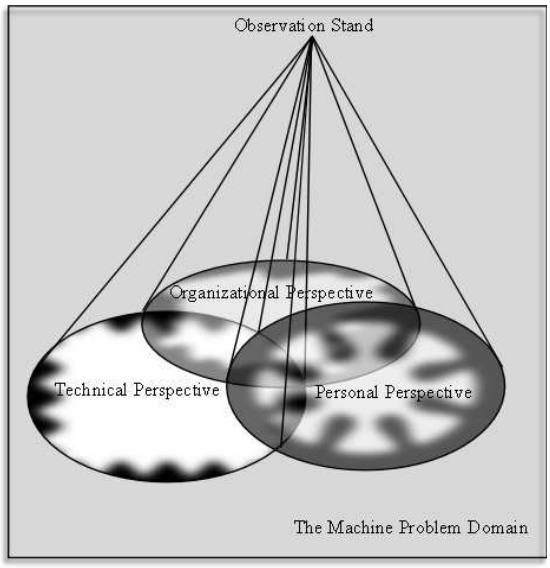


Figure 4. Common Membership Types

This grammatical sketch exposes new approach to define and articulate machine intelligence and the quotient. With this approach, one can fuse some variables and fix another for a controlled measurement. The linguistic method allows even a novice to understand what is measured. The tent is that unproductive intelligent machinery is a waste.

## References

[1] Auephanwiriyakul, S., & Keller, J. M. (2001). *A comparison of the linguistic Choquet and Sugeno fuzzy integral.* Paper presented at the 2001 IEEE International Fuzzy Systems.

[2] Auephanwiriyakul, S., Keller, J. M., & Gader, P. D. (2001). *Generalized Choquet fuzzy integral fusion.* Information Fusion(3), pp. 69–85.

[3] Bien, Z., Bang, W., Kim, D., & Han, J. (2002). *Machine intelligence quotient: its measurements and applications.* Fuzzy Sets and Systems, 127(1), pp. 3–16.

[4] Chen, S. J., Hwang, C. L., & Hwang, F. P. (Eds.). (1992). *Fuzzy multiple attribute decision making: Methods and Application* (Chen, S. J Hwang, C. L ed.). (Vol. 375). New York, NY: Springer-Verlag.

[5] Chen, C.-C., Lai, C.-M., Chen, T.-H., & Nien, H. (2005). *Statistical test of fuzzy hypothees using linguistic variables.* Journal of Industrial and Business Management, 1(1), pp. 11–22.

[6] Churchman, C. W. (1971). *The Design of inquiring systems.* New York, New York: Basic Books.

[7] Cotsaftis, M. (2000). *On definition of task oriented system intelligence.* Paper presented at the PerMIS: Workshop on Performance Metrics for Intelligent Systems, Gaithersburg, MD.

[8] Courtney, J. F., Croasdell, d. T., & Paradice, D. (1998). *Inquiring organization.* Foundation of Information Systems. Available: http://www.cba.uh.edu [2000, March 15].

[9] Dick, S. (2005). *Towards complex fuzzy logic.* IEEE Transaction of Fuzzy Systems, 13(3), pp. 405–414.

[10] Grabisch, M. (1995). *Anew algorithm for identifying fuzzy measures and its application to pattern recognition.* IEEE, pp. 145–150.

[11] Hasan, H. (1998). *Integrating IS and HCI using activity theory as a philosophical and theoretical basis.* Foundations of Information Theory. Available: http://www.cba.uh.edu [2000, May 15].

[12] Iovine, J. (1998). *Understanding neural networks.* Indanapolis, IN: Prompt Publications.

[13] Jennings, A. (1999). *Growing intelligence.* Jennings, Andrew. Available: ajennings@rmit.edu.au [2000, May 15].

[14] Ji, & Cheng. (2001). *Chance discovery in a BDI perspective of planning: Anticipation, participation and correlation:* Multi-Agent Systems Lab.

[15] Kant, I. (1787). *Critique of pure reason.* Macmillian Press Ltd. Available: www.arts.cuhk.edu/Philosophy/Kant/cpr-open.html [2002, July 20].

[16] Linstone, H. A. (1984). *Multiple perspectives for decision making: bridging the gap between analysis and action.* New York, NY: North-Holland.

[17] Lowell, B. E. (1995). *A Taxonomy of uncertainty.* Unpublished Dissertation, Portland State University, Portland. OR.

[18] Moses, D., Degani, O., Teodorescu, H.-N., Fiedman, M., & Kandel, A. (1999). *Linguistic Coordinate TRansformation for complex fuzzy sets.* Paper presented at the 1999 IEEE International Fuzzy Systems Conference Proceedings, Seoul, Korea.

[19] Moses, D., Teodorescu, H., Friedman, M., & Kandel, A. (1999). *Complex memebership grades with an application to the design of adaptive filters.* Computer Science of Journal of Moldova, 7(3(21)), pp. 253–283.

[20] Ramot, D., Friedman, M., Langholz, G., Milo, R., & Kandel, A. (2001). *On complex fuzzy sets.* IEEE International Fuzzy Conference, 1160–1163.

[21] Ramot, D., Milo, R., Friedman, M., & Kandel, A. (2002). *Complex fuzzy sets.* IEEE Transaction on Fuzzy Systems, 10(2), 171186.

[22] Sapp, J. C. (1987). *Eletricity demand forecasting in a changing reginonal context: The application of the multiple perspective concept to the prediction process.* Unpublished Dissertation, Portland State University, Portland, OR.

[23] Searle, J. (1983). *Intentionality: An essay in the phyilosophy of mind.* New york, NY: Cambridge University Press.

[24] Small, P. (1999). *Magical A-life avatars: A new paradigm for the Internet with examples in Macromedia's director.* Greenwich, CT: Manning.

[25] Tanaka, K. (1996). *An introduction to fuzzy logic for practical application* (Tak Niimura, Trans.). New York, NY: Springer.

[26] Tarr, S. C. (1990). *The knowledge transfer project: A multiple perspective investigation into the integration of a new technology within a business unit.* Unpublished Dissertation, University of Portland, Portland, OR.

[27] Turing, A. (1936). *On computable numbers: With an application to the enthscheidungsproblem.* Paper presented at the Proceedings of the London Mathemetical Society, London.

[28] Turing, A. (1950). *Computing machinery and intelligence.* Mind, 59(236), pp. 433–460.

[29] Ulinwa, I. C. (2003, 9-13-16-2003). *MIQ: Understanding a machine through multiple perspective analysis.* Paper presented at the PerMIS'03, Gaithersburg, MD.

[30] Wang, P. P. (2000). *Machine intelligence ranking.* Paper presented at the PerMIS: Workshop on Performance Metrics for intelligent Systems, Gaithersburg.

[31] Wolfram, S. (2002). *A new kind of science.* Champaign, IL: Wolfram Media, Inc.

[32] Zadeh, L. A. (1994). *Fuzzy logic, neural network, and soft computing.* Comunication of the ACM, 37(3), 77-84.

[33] Zeiber, A. R. (1996). *A system approach for rational decision making in potential strike situation.* Portland State University, Portland, OR.

V. C. I. Ulinwa,

Received March 26, 2007

Walden University School of Education, USA
E–mail: *iulinwa@waldenu.edu*

Computer Science Journal of Moldova, vol.**15**, no.1(43), 2007

# Aspect oriented programming and component assembly

A. Colesnicov L. Malahova

**Abstract**

The article describes an attempt to use the aspect oriented programming for construction of a graphical user interface from components. Aspects were applied to produce the glue code. Previously some doubts were expressed on possibility of such usage.

## 1 Introduction

The main efforts in development of complicated software products for scientific applications are applied to the creation of their computational engines that solve their target problems. Supplying a modern graphical user interface (GUI) for the resulting system is a different task that usually meets a shortage of resources. Our own experience with the Bergman symbolic computation system (SCS) and review of other systems illustrate this [1].

We investigate in this article a possibility to create a GUI for a given SCS semi-automatically from a set of ready-made components. The modern techniques of component programming (CP) give us a lot of useful approaches, e.g.: conceptions of black, grey, and white boxes; the notion of the *glue code* (the additional code that is necessary to assemble components together); lists of requirements for components, etc. Nevertheless, we found the existing CP frameworks not fully suitable: they provide less than we need in the automation of component assembly, and they provide much more than we need being oriented mainly towards distributed applications.

Aspect oriented programming (AOP) is a technique to add a new behavior to an existing program without changing its sources and even binaries. It is mostly used to handle cross-cutting concerns like logging or debugging. E.g., we need to add almost the same code in regularly selected places of the program to trace it. AOP concentrates templates of additional code and insertion points in *aspects.* Aspects are compiled separately, and the *code weaving* is performed during the execution of the program.

To apply AOP for the semi-automated assembly of a GUI from components, we noted that the glue code is regular and repeating, and that it can be generated from a formal description of the GUI. With AOP, we use an unchanged GUI template and unchanged components, and generate only aspects containing the glue code. We have checked this idea by implementing it.

Application of the AOP techniques to the CP was claimed in 1999 [2] but was not developed further that time, may be because that solution had supposed language extensions. Moreover, in 2003 C. Perez had published a note in his blog[1] under the title "Do aspects supersede components" that motivates the impossibility of such application. Our article describes a successful experiment in usage of the AOP for component assembly resolving therefore these doubts.

The article begins with a short review (Sec. 2–3) of existing techniques and frameworks of CP. See [3,4] for more details. Sec. 4 briefly describes the AOP. We describe in Sec. 5 our implementation of the idea formulated above. The next Sec. 6 discusses requirements to components. The article terminates with conclusions (Sec. 7) and acknowledgements.

[1] http://www.manageability.org/blog/archive/20030604%23do_aspects_supercede_components

## 2 Component programming: black-box techniques

Techniques of component adaptation may be roughly classified as *white-box, grey-box,* and *black-box* [5, p. 2–6].

At *white-box* reuse we actually need the full knowledge of component's internals. The most primitive kind of such reuse is the *copy-and-paste* technique known also as the *code scavenging*. Another technique is the *inheritance* usual at object-oriented programming. We meet here at the first time the important notion of the *glue code*, i.e., the code that assembles components together. At the inheritance, the glue code is represented by new methods in the subclasses of the original component.

*Grey-boxes* were thoroughly discussed in [5]. The following example is adapted from there: let we have a black-box component *Sort* that takes its input and produces the sorted output. We can replace this component. Suppose the initial component used the *Insertion sort* that is stable, i.e., keeps order of records with equal keys. We note this behavior during experimentation and use it in our programming of other parts of our system. Then we replace the component by another one that implements *Tree sort*. This algorithm is unstable, and we are in the wrong side. Therefore some knowledge of this component's internals is necessary. With *Sort*, we can return to the black-box model including stability in its output specifications but there are another situations where this is impossible, e.g., when a component depends on an external service [5, p. 3–5]. Grey boxes are used in most modern technologies.

*Black-boxes* hide all their internals; only input and output specifications are exposed. Due to their simplicity, this is the most comfortable model for assembly. There are many techniques to manipulate them. We present below several CP technologies based on black-box model.

## 2.1 Wrapping

This technique is also named *containment* when referred in COM technology (see 3.3 below). A *wrapper* can be defined as a container object that encapsulates a given black-box component and intercepts all its input and output. The simplest wrapper adapts the containing object's interface. More complicated wrappers can restrict or extend the functionality of their containing objects.

Wrappers perform additional calls during the interface tunnelling and can therefore lose in performance.

## 2.2 Superimposition

*Superimposition* was introduced by J. Bosch [6]. The component is included inside one ore more *layers* that intercept all messages to and from it. Each layer can convert a message in a passive object, analyze its contents and react in correspondence. Then some additional behavior is superimposed over the initial component's behavior.

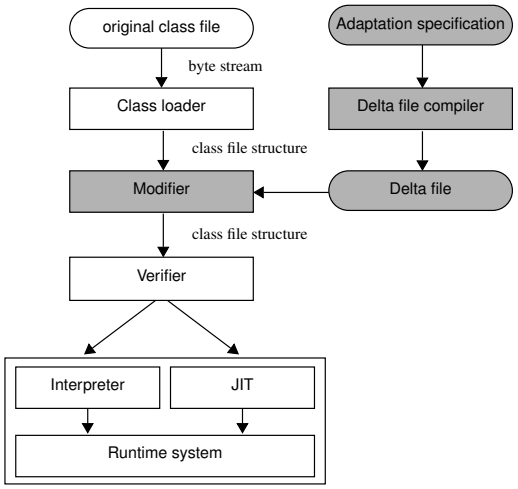


Figure 1. Binary component adaptation

### 2.3 Active interfaces and binary component adaptation

*Active interfaces* were proposed by G. Heineman [7]. An active interface decides whether to take action when a method is called. All interface requests pass two phases: the "before-phase" before the component performs any steps towards executing the request, and the "after-phase" when the component has completed all execution steps for the request. The active interface does not adapt component's behavior; instead, it enforces the necessary behavior at run-time.

*Binary component adaptation* [8] changes compiled (binary) component while it is loaded and was applied to Java. It is illustrated by Fig. 1 taken from [8, p. 2]. We see that *adaptation specification* is prepared separately and compiled separately by a *delta file compiler* to a *delta file*. A *modifier* is a single additional part of the Java virtual machine weaving the delta file with the original class.

Both these techniques were superseded by the *aspect oriented programming* (see Sec. 4 below).

## 3 Component programming: industrial component models

Attempts to define a set of standards for component implementation, customization, and composition led to such industrial component models as CORBA/CCM, JavaBeabs/EJB, COM, .NET.

### 3.1 CORBA/CCM

*Common Object Request Broker Architecture* (CORBA) and *CORBA Component Model* (CCM) were developed by OMG[2]. CORBA is an infrastructure that provides communication of distributed objects or components. The Object Request Broker (ORB) supports communication of components independently of their platforms or the methods of their implementation. All communicating components should be registered in an implementation repository. The components (objects)

[2]Object Management Group, http://www.omg.com

are represented by their *stubs* and *skeletons* that are their platform and language independent abstractions. To create stubs and skeletons, a specialized language called IDL (Interface Definition Language) is used. Instances of stubs look like local objects and accept method invocations from other objects. The actual target objects can be located elsewhere in the network and get these invocations through skeletons. Skeletons handle argument arrangement, actual method invocations and rearrangement of resulting values that are passed back to stubs.

## 3.2 JavaBeans/EJB

*JavaBeans* is a technology that is implemented for Java and permits to produce visual (graphical) components that are platform-independent, and reusable.

EJB are *Enterprize JavaBeans* introduced by SUN[3]. EJB are non-visual components of two kinds, *session beans* and *entity beans*. Session beans provide the communication of a client and a database. Entity beans represent the data from the databases and provide methods to manipulate these data. A session bean exists while a connection exists between a client and a database. An entity bean exists while the corresponding data exist in a database.

The EJB model allows the developer to implement business logic of applications and do not go into the things like transactions or security.

## 3.3 COM

Microsoft's *Common Object Model* (COM) defines a binary structure for interfaces between a COM objects (COM-compliant components) and their clients. There is a standard way to lay out virtual function tables (*vtables*) in memory, and a standard way to call functions through vtables. Components can be implemented in different languages. There exists Microsoft's IDL to define interfaces. Each object can provide several services. All services are registered in a system

[3] http://java.sun.com/developer/onlineTraining/Beans/index.html

registry using a *global unique identifier* (GUID). New implementations of existing objects use new GUIDs.

COM provides two methods of binary reuse, *containment* and *aggregation*. The first one is in fact wrapping: the object intercepts method calls and forwards them to its internal objects. In aggregation, a COM object exposes the services of another object as its own services.

### 3.4 .NET

The *.NET framework* (now version 3.0; regrettably, different versions are not compatible) is the latest platform from Microsoft that delivers components' services through Internet. A .NET application is composed of *assemblies*. An assembly contains compiled code and metadata. The manifest is included with each assembly and contains the assembly name, its version, the list of files, the list of dependencies, and the list of exported features.

## 4 Aspect-oriented programming

*Aspect-oriented programming* (AOP) generalizes, systematizes, and formalizes the *code weaving*. The code weaving was used in many circumstances like logging, tracing, debugging, security checking, etc. An example of early (1989–1990) use of the code weaving at the implementation of a debugger can be found in [9].

In AOP, the existing code is extended by *aspects*. An aspect contains *pointcuts* and *advices*. A pointcut is a template that defines *join points*. Each time the program execution passes a join point, the corresponding advice is executed. The advice code seems to be weaved with the original program code at the join point.

The *source code weaving* is a single possible solution in languages like C++. On the contrary, the Java binary code is well-defined and permits the *binary code weaving*. The AspectJ[4] implementation of

[4] http://www.eclipse.org/aspectj/

Aspect Java started with the source code weaving in 2001, then got the *bytecode weaver* at build time, and use the bytecode weaver at class loading since 2005.

The AOP is a popular and developing technique. Detailed descriptions and examples can be found elsewhere[5].

## 5 Implementation

We selected Java for portability, and Eclipse[6] with AspectJ plug-in as our implementation tool.

Suppose we have an engine that performs some symbolic calculations. The engine gets data for a computation session as a text file or several files. The engine does not interact with the user during calculation; it is, therefore, a true "black box" that takes data and produces results. We want to wrap the engine in a GUI that collects data, creates text files, runs the engine over these files, and shows results.

A GUI consists of the constant part and the variable part. The constant part contains the session management: storing data for each session, their modification, etc. We also found useful a notion of *environment*, or partially defined session [1]. Each session can be based on an environment where some data are already defined. The environment management is implemented like the session management.

Other features of the constant part of a GUI are possibilities to select one of several engines, to start external programs, to check collected data, to show help, etc.

Modules that enter the data form the variable part of a GUI. These modules depend on the problems solved by a particular engine.

During the assembly of a GUI its constant part is taken as the base. The developer prepares list of data and defines how they have to be entered in the GUI (by selection from several variants, by marking, by text editing, by 2D formula input, by entering parameters of a mathematical object using a wizard, etc.) Each possible method of

[5] http://en.wikipedia.org/wiki/Aspect_oriented_programming
[6] http://www.eclipse.org/

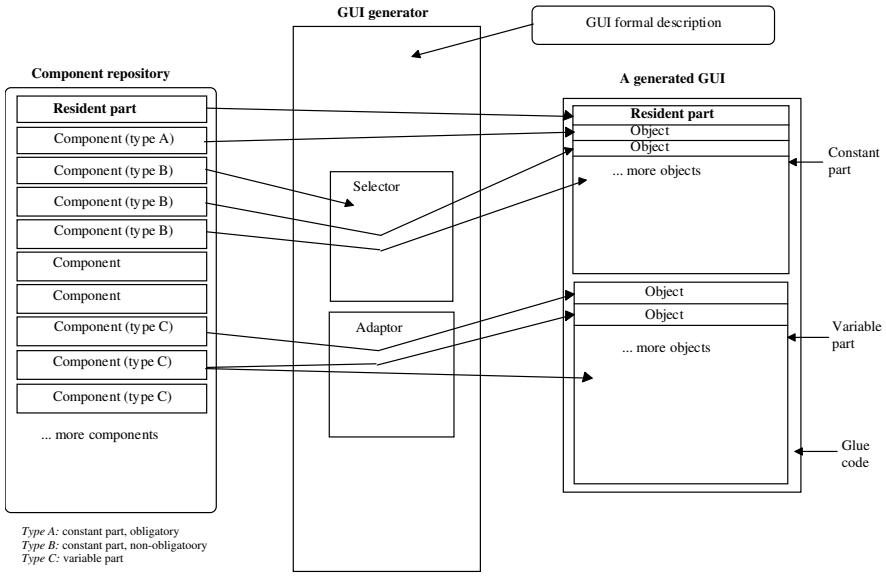


Figure 2. Generation of GUI in our project

the data input is implemented as a customizable component. The necessary modules pass the customization and are glued together with the constant part of the GUI.

The system consists therefore of a pre-implemented constant part, a set of data input components, and a GUI generator that adapts and assembles all parts together.

We already noted that the generated GUI contains the constant and the variable part. The constant part performs a lot of independent standard tasks (e.g., "Save session as..." or "Select engine"). We apply the CP techniques at the development of both parts of GUI. The constant part is also composed from several components and the *resident part*. There is a possibility to vary the constant part. Some components of the constant part are not obligatory. The GUI developer marks included components of the constant part in the list (default is "Include all").

There is some difference between components of the constant and variable GUI parts. The former do not need adaptation; the question is "to use or not to use". The latter are customizable through parametrization.

The generation of a GUI can be illustrated by Fig. 2.

The resident part contains, in particular, the *control center*. The control center registers all assembled components, collects and keeps data from data input components, and produces files for the engine.

The following techniques can be used to assemble applications from components:

- Manual assembly; the gluing code is written manually.
- Visual assembly in an IDE (Java Builder designer, etc.).
- Automated assembly.

The first two techniques are not suitable because they are oriented mainly towards a professional software developer. We selected semi-automated script-based assembly. At first it is necessary to plan the menu structure. The planned menu structure is fixed in an XML description. Using the XML description, a Java menu source is generated programmatically. For each menu item, its action is generated as an aspect. The menu, the aspects and the constant part of the GUI are weaved together resulting in a ready-made GUI.

Fig. 3 shows a simplified GUI that was generated using this technique. The opened dialog permits 2D input of polynomials. To assemble this component to the GUI, it was necessary to generate 2 lines of code in the Java menu source, and an aspect source of 11 lines (5 significant lines), in total 13(7) lines. The constant part of the GUI and the component sources were not changed at all.

## 6 Requirements to GUI components

Lüer and Rosenblum [10] define a component as a unit of independent deployment prepared for reuse that does not have persistent state. The

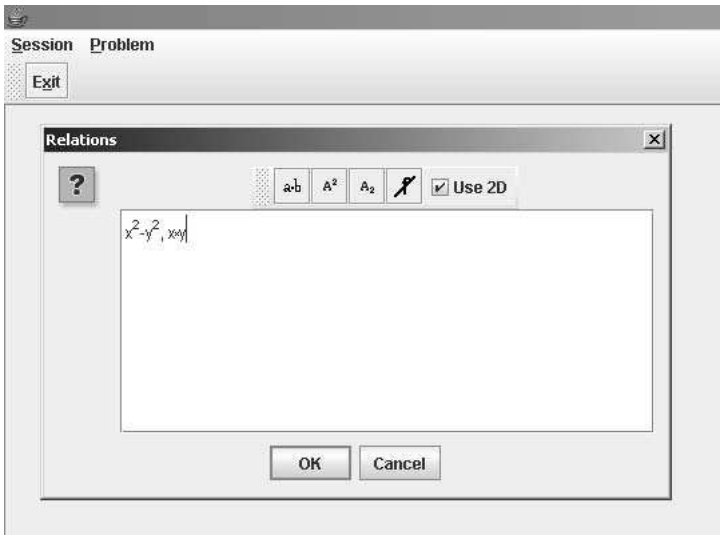


Figure 3. 2D input component in a generated GUI

latter means that a component is a set of classes and does not contain objects.

The following seven requirements of component development were formulated in [10].

1. We should follow the common principles of modular design. In particular, the private and public parts of a component should be separated.

2. Component should be self-descriptive.

3. Component interfaces should be accessed within a global name space through unique names.

4. Development process has two parts, component development and application composition.

5. Application development consists of the component composition and the implementation of additional functionality that is not available in components.

6. We should provide multiple views: a component developer view, a composition view, a type view, an instance view, an overall structure view, etc.

7. The best component reuse is achieved through reuse by reference.

Now we discuss the application of these seven requirements in our case.

1. **Modular design.** The public part of a component should contain public interface definitions, and, optionally, installation tools and self-description. The private part contains implementation (set of classes) and resources (e.g., graphic or help files).

   In [10], installation tools and self-description are exposed as obligatory subparts of component's public part. Installation means providing instances of the classes implemented into the component (objects) for component's clients. It is straightforward in the simplest case; however, a more complicated component may internally select from different implementations of the same published interface.

   In our case, separation of public and private parts will be strictly adhered. Installation means for us parametrization and inclusion in the generated GUI, i.e., gluing. This is made using aspects as described above in Sec. 5.

2. **Self-description.** It is used in a limited way in industrial component models like Java Beans, COM, and .NET. Lüer and Rosenblum [10, p. 4] define five levels of self-description:

   1. The *syntactic* level. It provides signatures of abstract data types that are provided by the component or required by it. A special language (IDL) can be used for such descriptions. We saw that this is a common feature in many CP platforms.

2. The *behavioral* level. It provides semantic description of data types. The semantic description can be informal, semi-formal, or even formal.
3. The *synchronization* level. It provides information that permits cooperating components to resolve concurrency problems.
4. The *QoS,* or *non-functional,* level. QoS (*Quality of Service*) is the usual term for all non-functional specifications like the volume of used memory, the response delay, the result precision, etc.
5. The *non-technical* level. It provides other information like price, contact address, support phone, gained official certifications, etc.

Self-description is less necessary as our components will be used by the GUI generator in programmed manner. The corresponding parts of aspects are in fact self-description in the syntactic level.

3. **Global naming scheme.** We will use our components inside a monolithic system; therefore, we have to use unique names.

4. **Two-stage development process.** In our case, component development and component usage are strictly separated in time. The assembly of a GUI is executed by a different developer, probably by a mathematician that programmed a calculation engine and wants the GUI for it.

5. **Functionality not present in components.** The constant part of GUI provides such functionality in our case. Components are dialogs or wizards to enter one or more session parameters.

6. **Multiple views.** In our case, the component development and their composition are strictly separated in time. The instance view may be useful during the GUI generation.

7. **Reuse by reference.** This principle is applicable in the case when the components have different sources and need maintenance, e.g., distant update. Such option is not supposed in our development, but we could provide it later. If the resulting system became widely used, the existence of a central component repository may be useful.

# 7 Conclusions

There exists many classifications of component assembly paradigms. One such classification was proposed by a research group[7] at the University of Waterloo, Canada. The classification is as follows:

**Design Patterns** describe how to assemble objects basing on the separation of concerns. Design patterns capture design experience "in-the-small". The viewpoint approach to software design helps the designer to isolate many design pattern constructs.

**Frameworks** are semi-finished software architectures for a specific application domain, and they attempt to capture design experience "in-the-large". Frameworks represent the highest level of reusability currently found in software design and implementation. Design patterns and frameworks are related, in that design patterns are normally used to assemble components in a framework.

We have a semi-finished GUI (constant part) and a set of components. Therefore our solution can be classified as *a framework* that is based on the AOP and automates the CP. This is a successful attempt to widen the area of the AOP towards the realm of components, at least for adaptable monolithic applications.

[7] http://csg.uwaterloo.ca/program.html

## Acknowledgements

The work was supported by the INTAS grant Ref. Nr. 05–104–7553 "Interface generating toolkit for symbolic computation systems". 2D input component was adapted by dr. Ludmila Burtseva.

## References

[1] S. Cojocaru, L. Malahova, A. Colesnicov. *Interfaces to symbolic computation systems: reconsidering experience of Bergman.* // Computer Science Journal of Moldova, vol. **13**, no. 2(28), 2005, p. 232–244.

[2] K. Lieberherr, D. Lorenz, M. Mezini. *Programming with Aspectual Components.* College of Computer Science, Northeastern University, Boston, MA, Report #NU–CCS–99–01, March, 1999.

[3] I.-G. Kim, D.-H. Bae. *Dimensions of Composition Models for Supporting Software System Evolution.* Technical report CS/TR–2005–244, Dept. of Electrical Engineering and Computer Science, KAIST, Daejon, Korea, December 7, 2005.

[4] A.J.A. Wang, K. Qian. *Component-oriented programming*, A Wiley-Interscience publication, 2005, ISBN 0–471–64446–3.

[5] M. Büchi, W. Weck. *A Plea for GreyBox Components.* Turku Centre for Computer Science, TUCS Technical Report No. 122, August 1997, ISBN 952-12-0047-2, ISSN 1239-1891.

[6] J. Bosch. *Composition through Superimposition.* / In: Object-Oriented Technology – ECOOP'97 Workshop Reader, J. Bosch, S. Mitchell (eds.), LNCS 1357, Springer-Verlag, 1997.

[7] G. Heineman. *A model for designing adaptable software components.* / Proceedings, 22$^{nd}$ International Conference on Computer Software and Applications Conference (COMPSAC), Vienna, Austria, Aug. 1998, p. 121–127.

[8] R. Keller, U. Holzle. *Binary Component Adaptation.* Department of Computer Science, University of California, Santa Barbara, CA 93106, Technical Report TRCS97-20, December 3, 1997.

[9] A. Colesnicov, L. Malahona. *Some interface problems between a compiler and a conversational debugger.* Buletinul Academiei de Ştiinţe a Republicii Moldova, Matematica, Nr. 3(6), 1991, p. 12–20, ISSN 0236–3089. – In Russian.

[10] C. Lüer, D. Rosenblum. *WREN – An Environment for Component-Based Development.* Department of Information and Computer Science, University of California, Irvine, CA 92697, Technical Report #00–28, September 1, 2000.

Received April 10, 2007

A. Colesnicov, L. Malahova,

Institute of Mathematics and Computer Science,
5 Academiei str.
Chişinău, MD–2028, Moldova.
E–mail: *kae@math.md*, *mal@math.md*

Computer Science Journal of Moldova, vol.**15**, no.1(43), 2007

# The Pico's formula Generalization

Sergiu Cataranciuc, Marina Holban

**Abstract**

The Pico formula generalizations are obtained for area calculation of a polygon $P$ through the determination of special nodes of the network in which this $P$ is placed. The case of the polygon with rational coordinates of its vertexes is examined, as well as the case of the polygon with holes. In the case of three-dimensional space a formula of volume calculation for some polyhedrons, such as prism and tetrahedron is presented. On the basis of theoretic outcomes an algorithm that can be applied in calculation for areas of plane figure is elaborated.

## 1 The Pico's formula

The determination problem of some efficient formulas for areas calculation of some plane figures classes presents a certain interest from both theoretical and practical point of view. In general case, the integral calculus come to help that will make this problem quite difficult, both from the point of view of function construction that describes the figure frontier and of subsequent calculations that must be effectuated. Thus the study of some special plane figure classes, in particular polygons, becomes very important.

Let us consider one arbitrary polygon $P$ in the plane the vertexes of which are the nodes of a rectangular network $D$. Network $D$ is determined by classes of parallel to axes $OX$ and $OY$ lines. The crossing points of lines are called the network nodes.

Depending on the structure of network $D$ the formulas for efficient calculation of area of the polygon $P$ are known. The result obtained by the Austrian mathematician G. Pico in 1899 is considered to be

among the first results in this direction. In his work G. Pico studies the network case, determined by classes of parallel to axes $OX$ and $OY$ straight lines in which the distance between any two neighbour straight lines is equal to one. Such a network will be called unitary network.

**Theorem 1.1. [1].** *The area of any polygon $P$, constructed in a unitary network $D$ is calculated by the following formula:*

$$S(P) = i + \frac{b}{2} - 1, \tag{1}$$

*where $b$ represents the number of points of the network situated on the frontier of the polygon, and $i$ represents the number of points of the network that belong to interior of this polygon.*

Let's illustrate the formula from Theorem 1.1 for polygons $P_1$ and $P_2$ from Figure 1.

Polygon $P_1$: On the one hand, as it can be observed the polygon $P_1$ is formed from 4 complete squares and 8 triangles (halves of squares) of the unitary network. Thus $S(P_1) = 4 + 8 \cdot \frac{1}{2} = 8$. Applying the formula (1), taking into consideration that $i = 1$ and $b = 16$, we obtain the same result:

$$S(P_1) = i + \frac{b}{2} - 1 = 1 + \frac{16}{2} - 1 = 8.$$

Polygon $P_2$: The polygon $P_2$ can easily be reduced to some simple polygons, trapeziums or triangles, the area of which is easily calculated by the formula:

$$\begin{aligned} S(P_2) &= S(ALFH) - S(AIH) - S(DEF) - S(HGF) - S(BLDC) = \\ &= AH \cdot HF - \frac{AH \cdot h_{AH}}{2} - \frac{HF \cdot h_{FH}}{2} - \frac{BL + CD}{2} \cdot LD = \\ &= 4\dot{5} - \frac{4 \cdot 1}{2} - \frac{5 \cdot 1}{2} - \frac{3 \cdot 1}{2} - \frac{2 + 1}{2} \cdot 1 = \frac{25}{2} \ . \end{aligned}$$

On the other hand, applying formula (1) we obtain:

$$S(P_2) = i + \frac{b}{2} - 1 = 8 + \frac{11}{2} - 1 = \frac{25}{2} \ .$$

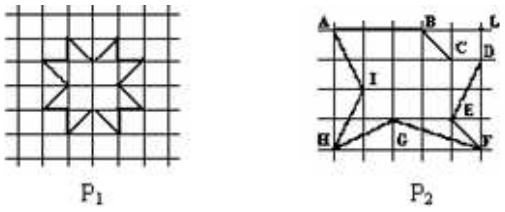


Figure 1.

## 2 The generalization of Pico's Formula in the rational polygons

Let us consider $P$ a polygon, vertexes of which have rational coordinates. This kind of polygons will be called rational polygon. If the polygon has $k+1$ vertexes, than we form the increasing series of abscissas

$$x_1, \quad x_2, \quad \ldots, \quad x_k, \quad x_{k+1}$$

and of ordinates

$$y_1, \quad y_2, \quad \ldots, \quad y_k, \quad y_{k+1}$$

of the polygon $P$ vertexes. We mark

$$d_i^x = x_{i+1} - x_i, \quad \forall i = \overline{1,k} \ ,$$

$$d_i^y = y_{i+1} - y_i, \quad \forall i = \overline{1,k} \ .$$

Without losing from the generality we consider that $d_1^x, \quad d_2^x, \quad \ldots, \quad d_k^x$ are distinct, and $d_1^y, \quad d_2^y, \quad \ldots, \quad d_k^y$ as well. Let us form the multitude $D = \{d_1^x, \quad d_2^x, \quad \ldots, \quad d_k^x, \quad d_1^y, \quad d_2^y, \quad \ldots, \quad d_k^y\}$. One rational number $\alpha$ will be named *divisor* of the rational number $\beta$ if $\frac{\beta}{\alpha} \in \mathbb{Z}$ (Here $\mathbb{Z}$ represents the multitude of integer numbers). In the case when $\alpha, \quad \beta, \quad \gamma$ are 3 rational numbers and $\frac{\alpha}{\gamma} \in \mathbb{Z}, \quad \frac{\beta}{\gamma} \in \mathbb{Z}$, than

we will say that $\gamma$ is the common divisor of numbers $\alpha$ and $\beta$. It is obviously that for any multitude of rational numbers there exists a common divisor. Let us denote the common divisor of the multitude $D$ of elements by $d'$.

Let us trace in space $\mathbb{R}^2$ parallel to axes $OX$ and $OY$ straight lines so that the distance between each 2 neighbour straight lines to be equal to $d'$. As a result we obtain a network marked by $D(d')$. Obviously, the vertexes of the rational polygon $P$ are situated in the nodes of this network.

**Theorem 2.1.** *If the vertexes of rational polygon $P$ are situated in the nodes of the network $D(d')$, then the area of this polygon is calculated by the formula:*

$$S(P) = \left( i' + \frac{b'}{2} - 1 \right) \cdot (d')^2 \ , \tag{2}$$

*where $i'$ is the number of nodes of network $D(d')$ which belong to the interior of polygon $P$ and $b'$ - number of nodes on the frontier of $P$.*

**Proof:** We'll prove the affirmation of the theorem by the mathematical induction on $n$ vertexes of polygon $P$. When $n = 3$ polygon is a triangle $ABC$ with certain rational coordinates of vertexes: $A = (x_A, \quad y_A)$, $B = (x_B, \quad y_B)$, $C = (x_C, \quad y_C)$. Let us pass to another coordinate system $OX^*Y^*$, in which $x^* = \frac{x}{d'}, y^* = \frac{y}{d'}$. In this system vertexes of the triangle will have coordinates: $A^* = \left(\frac{x_A}{d'}, \quad \frac{y_A}{d'}\right)$, $B^* = \left(\frac{x_B}{d'}, \quad \frac{y_B}{d'}\right)$ and $C^* = \left(\frac{x_C}{d'}, \quad \frac{y_C}{d'}\right)$. Let us calculate the triangle's area in the coordinate system $OX^*Y^*$:

$$S_{\triangle A^*B^*C^*} = \left| \frac{1}{2} \cdot \begin{vmatrix} \frac{x_A}{d'} & \frac{y_A}{d'} \\ \frac{x_B}{d'} & \frac{y_B}{d'} \end{vmatrix} + \frac{1}{2} \cdot \begin{vmatrix} \frac{x_B}{d'} & \frac{y_B}{d'} \\ \frac{x_C}{d'} & \frac{y_C}{d'} \end{vmatrix} + \frac{1}{2} \cdot \begin{vmatrix} \frac{x_C}{d'} & \frac{y_C}{d'} \\ \frac{x_A}{d'} & \frac{y_A}{d'} \end{vmatrix} \right| =$$

$$= \left| \frac{1}{2} \cdot \frac{1}{d'} \cdot \frac{1}{d'} \cdot \begin{vmatrix} x_A & y_A \\ x_B & y_B \end{vmatrix} + \frac{1}{2} \cdot \frac{1}{d'} \cdot \frac{1}{d'} \cdot \begin{vmatrix} x_B & y_B \\ x_C & y_C \end{vmatrix} + \frac{1}{2} \cdot \frac{1}{d'} \cdot \frac{1}{d'} \cdot \begin{vmatrix} x_C & y_C \\ x_A & y_A \end{vmatrix} \right| =$$

$$= \frac{1}{2} \cdot \left( \frac{1}{d'} \right)^2 \cdot \left| \begin{vmatrix} x_A & y_A \\ x_B & y_B \end{vmatrix} + \begin{vmatrix} x_B & y_B \\ x_C & y_C \end{vmatrix} + \begin{vmatrix} x_C & y_C \\ x_A & y_A \end{vmatrix} \right| = \frac{1}{(d')^2} \cdot S_{\triangle ABC} \ .$$

The network $D(d')$ nodes in the coordinate system $OX^*Y^*$ are the coordinates in integer numbers. According to Pico's formula

$$S_{\triangle A^*B^*C^*} = i' + \frac{b'}{2} - 1,$$

where $i'$ is the number of nodes of the network $D(d')$ which belong to the interior of triangle $A^*B^*C^*$, and $b'$ – is the number of nodes of this network situated on the frontier of $P$. Thus

$$S_{\triangle A^*B^*C^*} = \frac{1}{(d')^2} \cdot S_{\triangle ABC} \ .$$

Hence

$$S_{\triangle ABC} = S_{\triangle A^*B^*C^*} \cdot (d')^2$$

or

$$S_{\triangle ABC} = \left( i' + \frac{b'}{2} - 1 \right) \cdot (d')^2 \ .$$

Thus the induction base is proved.

Let's admit that the theorem affirmation is true for any polygon $P$ with the number of vertexes smaller than $n$, $n \geq 4$. Let us analyse a polygon with $n$ vertexes from network $D(d')$. Let $A_1, \quad A_2, \quad \ldots, \quad A_n$ be the vertexes of this polygon, described clockwise (CW). Without losing the generality let us consider that $A_{n-1}, \quad A_n, \quad A_1$ are non-collinear points in plan. Let us connect $A_{n-1}$ with $A_1$. We obtain the triangle $A_1A_{n-1}A_n$ and $A_1A_2 \ldots A_{n-1}$ polygon with $(n-1)$ vertexes. Depending on the position of the initial polygon vertexes in plan we obtain

$$S(P) = S_{A_1 \ldots A_{n-1}} + S_{\triangle A_1 A_{n-1} A_n}$$

or

$$S(P) = S_{A_1 \ldots A_{n-1}} - S_{\triangle A_1 A_{n-1} A_n}$$

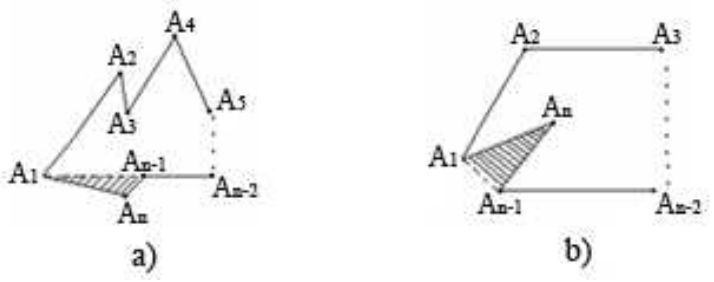


Figure 2.

(see Figure 2, cases a) and b)).

Let $i'$ be the number of nodes of the network $D(d')$ that are the interior points of the polygon $P = A_1A_2 \dots A_{n-1}A_n$, $i'_{n-1}$ – the number of nodes of the polygon $A_1A_2 \dots A_{n-1}$ with $n-1$ vertexes, and $i'_{\triangle}$ – the number of nodes of the triangle $A_1A_{n-1}A_n$. Let $b'$, $b'_{n-1}$, $b'_{\triangle}$ be the number of nodes of the network $D(d')$ that are situated on the frontier of the mentioned polygons. Let $b''_{A_1A_{n-1}}$ also be the number of nodes of the network, that are interior points of the segment $[A_1A_{n-1}]$, and let $b''_{n-1}$ and $b''_{\triangle}$ be the number of nodes that respectively belong to the broken lines $A_1A_2 \dots A_{n-1}$ and $A_{n-1}A_nA_1$. Thus, in the polygon from the Figure 2 a) we have the relations

$$i' = i'_{n-1} + i'_{\triangle} + b''_{A_1A_{n-1}} ,$$

$$b' = b'_{n-1} + b'_{\triangle} - 2 \cdot b''_{A_1A_{n-1}} - 2 = b''_{n-1} + b''_{\triangle} - 2 .$$

In the case of polygon from figure 2 b) we obtain

$$i' = i'_{n-1} - i'_{\triangle} - b''_{\triangle} + 2$$

(number 2 corresponds to the vertexes $A_1$ and $A_{n-1}$ that are included into the number $b''_{\triangle}$ but aren't interior for the polygon)

$$b' = b'_{n-1} + b''_{\triangle} - b''_{A_1A_{n-1}} - 2 .$$

Let us pass to the calculation of areas for the polygons from the Figure 2 a) and b).

For the case represented in Figure 2 a), considering mathematical induction method we obtain

$$S(P) = S_{\triangle A_1 A_{n-1} A_n} + S_{A_1 \ldots A_{n-1}} =$$

$$= \left( i'_{n-1} + \frac{b'_{\triangle}}{2} - 1 \right) \cdot (d')^2 + \left( i'_{n-1} + \frac{b'_{n-1}}{2} - 1 \right) \cdot (d')^2 =$$

$$= \left( i'_{\triangle} + i'_{n-1} + \frac{(b'_{\triangle} + b'_{n-1} - 2 \cdot b''_{A_1 A_{n-1}} - 2) + 2 \cdot b''_{A_1 A_{n-1}} + 2}{2} - 2 \right) \cdot (d')^2 =$$

$$= \left( i'_{\triangle} + i'_{n-1} + b''_{A_1 A_{n-1}} + \frac{b'_{\triangle} + b'_{n-1} - 2 \cdot b''_{A_1 A_{n-1}} - 2}{2} - 1 \right) \cdot (d')^2 =$$

$$= \left( i' + \frac{b'}{2} - 1 \right) \cdot (d')^2 .$$

Analogically, for the polygon represented in Figure 2 b) we obtain

$$S(P) = S_{A_1 \ldots A_{n-1}} - S_{\triangle A_1 A_{n-1} A_n} =$$

$$= \left( i'_{n-1} + \frac{b'_{n-1}}{2} - 1 \right) \cdot (d')^2 - \left( i'_{\triangle} + \frac{b'_{\triangle}}{2} - 1 \right) \cdot (d')^2 =$$

$$= \left( i'_{n-1} - i'_{\triangle} + \frac{b'_{n-1} - b'_{\triangle}}{2} - 1 \right) \cdot (d')^2 =$$

$$= \Bigg( \Bigg( \Bigg( \underbrace{i^{'}_{n-1} - i^{'}_{\triangle} - b^{''}_{\triangle}}_{i^{'}} + 2 \Bigg) + b^{''}_{\triangle} - 2 \Bigg) +$$

$$+ \frac{\left( \overbrace{b^{'}_{n-1} + b^{''}_{\triangle} - b^{''}_{A_1 A_{n-1}} - 2}^{b^{'}} \right) - b^{''}_{\triangle} + b^{''}_{A_1 A_{n-1}} + 2 - b^{'}_{\triangle}}{2} \Bigg) \cdot$$

$$\cdot (d^{'})^2 = \left( i^{'} + \frac{b^{'}}{2} - 1 + b^{''}_{\triangle} - \frac{b^{''}_{\triangle} + b^{'}_{\triangle} - b^{''}_{A_1 A_{n-1}}}{2} \right) \cdot (d^{'})^2 =$$

$$= \left( i^{'} + \frac{b^{'}}{2} - 1 + b^{''}_{\triangle} - \frac{b^{''}_{\triangle} + b^{''}_{\triangle}}{2} \right) \cdot (d^{'})^2 = \left( i^{'} + \frac{b^{'}}{2} - 1 \right) \cdot (d^{'})^2 \ .$$

Theorem is proved.

Let us illustrate this demonstration for the case of a polygon $P$ with the vertexes $A(\frac{1}{2}, \ \frac{1}{3})$, $B(\frac{2}{3}, \ \frac{5}{6})$, $C(\frac{2}{3}, \ \frac{1}{2})$ and $D(\frac{4}{3}, \ \frac{2}{3})$. It is easy to calculate $d^{'} = \frac{1}{6}$. Thus the polygon $P$ can be placed in the network $D\left(\frac{1}{6}\right)$ (see Figure 3).

In this case we obtain $i^{'} = 1$, $b^{'} = 5$ and, thus

$$S_{ABCD} = \left( 1 + \frac{5}{2} - 1 \right) \cdot \left( \frac{1}{6} \right)^2 = \frac{5}{72} \ .$$

On the other hand, the polygon $ABCD$, being divided in 2 triangles $ABC$ and $ACD$ has the area

$$S_{\triangle ABC} = \frac{1}{2} \bmod \begin{vmatrix} x_1 - x_3 & y_1 - y_3 \\ x_2 - x_3 & y_2 - y_3 \end{vmatrix} = \frac{1}{2} \bmod \begin{vmatrix} -\frac{1}{6} & -\frac{1}{6} \\ 0 & \frac{2}{6} \end{vmatrix} = \frac{1}{2} \cdot \frac{2}{36} = \frac{1}{36},$$

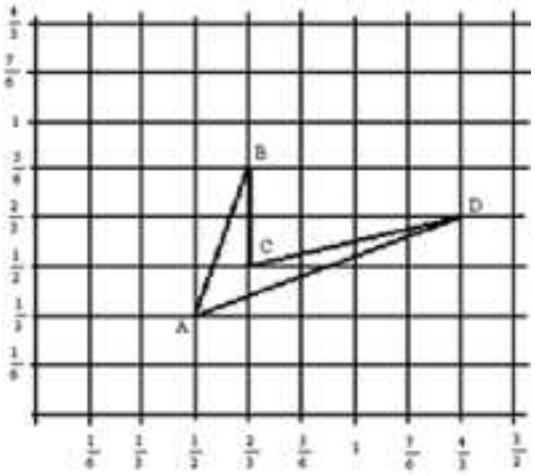


Figure 3.

$$S_{\triangle ACD} = \frac{1}{2} \bmod \begin{vmatrix} -\frac{5}{6} & -\frac{2}{6} \\ -\frac{4}{6} & -\frac{1}{6} \end{vmatrix} = \frac{1}{2} \cdot \frac{3}{36} = \frac{3}{72} \ .$$

So $S_{ABCD} = S_{\triangle ABC} + S_{\triangle ACD} = \frac{1}{36} + \frac{3}{72} = \frac{5}{72}$, which corresponds to the calculation made in concordance with theorem's formula.

By analogy, we will say that real number $\beta$ is a divisor of the real number $\alpha$, if $\frac{\alpha}{\beta} \in \mathbb{Z}$.

**Consequence:** If $P$ is a polygon, the vertexes of which have real coordinates, and $d^{'}$ is a the common divisor of these coordinates than $P$ can be placed in the network $D(d^{'})$, and its area is calculated by the formula

$$S(P) = \left( i^{'} + \frac{b^{'}}{2} - 1 \right) \cdot (d^{'})^2 \ ,$$

where $i'$ represents the number of nodes of network $D(d^{'})$ which belongs to the polygon interior, and $b^{'}$ - the number of vertexes situated on the frontier of $P$.

On the basis of those mentioned above let us elaborate a calculation algorithm of a polygon area according to the studied formula, in the conditions when the common divisor $d^{'}$ exists.

Let us admit that $A_1, \quad A_2, \quad \ldots, \quad A_n$ are the polygon $P$ vertexes, described clockwise.

Description of the algorithm

Step I. Let us draw the network $D(d^{'})$.

Step II. Let $a$ and $b$ represent the smallest and biggest values among the abscissas of the vertexes $A_1, \quad A_2, \quad \ldots, \quad A_n$ of polygon $P$, and $c$ and $d$ – the smallest and biggest values among the ordinates of these vertexes.

Step III. Let us construct the rectangle $D$, determinated by the straights $x = a$, $x = b$, $y = c$ and $y = d$. Obviously, in the described conditions, the polygon $P$ will be situated in the interior of the rectangle $D$.

Step IV. Let us construct the ordinate series $x_1 = a$, $x_2 = x_1 + d^{'}$, $x_3 = x_2 + d^{'}, \ldots, x_l = b$ where $l = \frac{b-a}{d^{'}} + 1$.

Step V. We trace the straight $X = x_i$ for $\forall i = \overline{1,n}$ . We study the knots of the network $D(d^{'})$ which belong to the rectangle $D$ and are situated on the straight $X = x_i$.

Let's consider $V$ one of these knots:

a) If there exists a segment $[A_j, \quad A_{j+1}]$, $j = \overline{1,n}$ (it's considered $A_{n+1} = A_1$ ) that contains point $V$, then this, which is a knot on the frontier of $P$, will be taken up in the calculation of number $b^{'}$.

b) If the condition a) is not realized, then we trace from the point $V$ a semi-straight, parallel to the positive direction of axis $OX$. We calculate the number $L$ of intersections of this semi-straight with frontier of polygon $P$. If the intersection is realized in some vertex $A_j$, $j = \overline{1,n}$ of the polygon, then

such intersection will be taken into consideration in the calculation of $L$ in the case when the neighbouring vertexes $A_j$ and $A_{j+1}$ are situated on different sides of the semi-straight. The knot $V$ of the network $D(d^{'})$, belongs to the interior of the polygon $P$ if and only if the number $L$ is odd.

As a result of application of items a) and b) for each straight $X = X_i$, $i = \overline{1,n}$ we'll obtain the values $i^{'}$ and $b^{'}$ . Thus, applying respective formula we calculate the polygon area.

Analyzing the described algorithm, we make sure of correctness of the following result.

**Theorem 2.2** *The area of the polygon $P$ situated in one network $D(d^{'})$ can be calculated in time $O(N^2)$, where $N$ is the number of vertexes of the polygon $P$.*

## 3 Polygons with holes

According to those described above, it's fascinating the fact that similarly to the formula exposed in the Theorem 1.1, there is the subtraction formula of the area of polygon $P_g$ with holes, built in a single network

$$S(P_g) = i + \frac{b}{2} - \chi(P_g) + \frac{1}{2}\chi(\delta P_g), \tag{3}$$

where $b$ represents the number of network knots which are situated on the frontier of the polygon $P_g$, but $i$ represents the number of network knots which belong to the interior of this polygon, $\chi(P_g) = 1 - n$ – Euler formula for the considered polygon with holes ( $n$ – the number of polygon holes), but $\delta P_g$ denotes the frontier of this ($\chi(\delta P_g) = b - M_b$, $M_b$ – the number of edges belonging to the frontier of polygon $P_g$) [1].

Let's illustrate the formula (1) for the polygon from Figure 4.

The polygon $P_g$ can easily be reduced to some simpler polygons, the area of which is easily determined just so:

$$\begin{aligned} S(Pg) =& S(ADVZ) - S(ABC) - S(DEF) - S(FGV) - S(HIZ) - \\ & - S(ALM) - S(BNEO) - S(PRST) = \end{aligned}$$

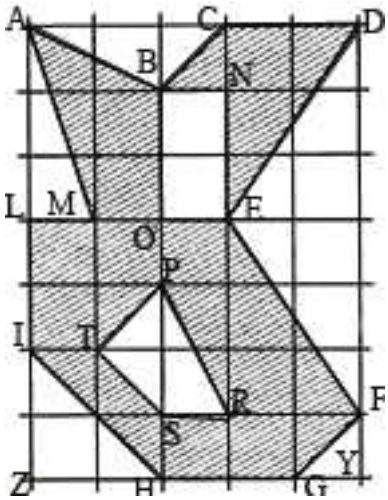


Figure 4.

$$= AZ \cdot ZV - \frac{AC \cdot h_{AC}}{2} - \frac{DF \cdot h_{DF}}{2} - \frac{FV \cdot VG}{2} - \frac{HZ \cdot ZI}{2} - \frac{AL \cdot LM}{2} -$$
$$- BO \cdot OE - \left( \frac{PS \cdot h_{PS}}{2} + \frac{SR \cdot h_{SR}}{2} \right) =$$

$$= 7 \cdot 5 - \frac{3 \cdot 1}{2} - \frac{6 \cdot 2}{2} - \frac{1 \cdot 1}{2} - \frac{2 \cdot 2}{2} - \frac{3 \cdot 1}{2} - 2 \cdot 1 - \left( \frac{2 \cdot 1}{2} + \frac{1 \cdot 2}{2} \right) = \frac{39}{2}.$$

Elsewhere, applying formula (1) and taking up that $i = 8$, $b = 23$, $\chi(P_g) = 1 - 2 = -1$ and $\chi(\delta P_g) = b - M_b = 23 - 25 = -2$, we obtain the same result:

$$S(P_g) = 8 + \frac{23}{2} - (-1) + \frac{1}{2} \cdot (-2) = \frac{39}{2}$$

We can easily make sure that, the area of the rational polygon $P_g$ with holes, the vertexes of which belong to the network $D(d^{'})$ (previ-

ously described), are calculated by formula

$$S(P_g) = \left( i^{'} + \frac{b^{'}}{2} - \chi(P_g) + \frac{1}{2} \cdot \chi(\delta P_g) \right) \cdot (d^{'})^2 \ ,$$

where $i^{'}$ – number of knots of network $D(d^{'})$ which belong to the interior of polygon $P_g$ with holes, $b^{'}$ – number of knots which are situated on the frontier of $P_g$, but $\chi(P_g)$ – Euler formula for the considered polygon with $n$ holes, $\chi(\delta P_g)$ represents the Euler formula of this frontier.

We'll illustrate those affirmed, in case of polygon $P_g$ from Figure 5. It's easily determined that $d^{'} = \frac{1}{3}$. As sequel, the polygon $P_g$ with holes can be placed in network $D\left(\frac{1}{3}\right)$.

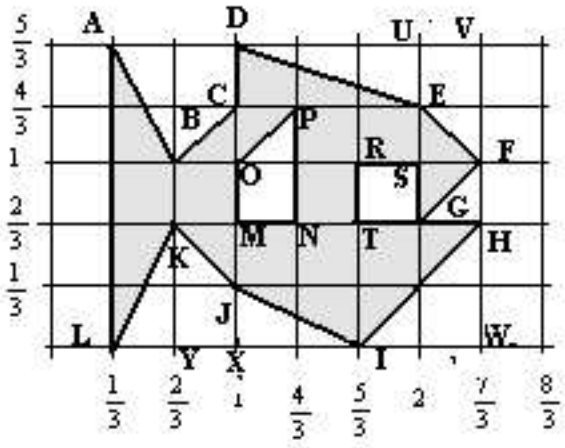


Figure 5.

In this case, we have $i^{'} = 3$, $b^{'} = 25$ and $\chi(P_g) = 1 - n = 1 - 2 = -1$, $\chi(\delta P_g) = b^{'} - M_{b'} = 25 - 26 = -1$ , as continuation, the polygon has the area

$$S(P_g) = \left( 3 + \frac{25}{2} - (-1) + \frac{1}{2} \cdot (-1) \right) \cdot \left( \frac{1}{3} \right)^2 = \frac{32}{18} \ .$$

Dividing the given polygon into simpler polygons, it's easily verified that we obtain the same result just so:

$$S(P_g) = S(AVWL) - 2\cdot(S(LKY) + S(KJXY)) - S(DEU) - S(UEFV) - S(FGH) - S(HIW) - S(IJX) - S(OMPN) - S(RSGT) =$$

$$= AL \cdot LW - 2 \cdot \left( \frac{LY \cdot YK}{2} - \frac{KY + JX}{2} \cdot XY \right) - \frac{DU \cdot UE}{2} - \frac{UE + VF}{2} \cdot UV - \frac{FH \cdot GH}{2} - \frac{HW \cdot WI}{2} - \frac{IX \cdot XJ}{2} - \frac{OM + PN}{2} \cdot \frac{1}{3} - RS^2 =$$

$$= \frac{5}{3} \cdot \frac{6}{3} - 2 \cdot \left( \frac{\frac{1}{3} \cdot \frac{2}{3}}{2} + \frac{\frac{2}{3} + \frac{1}{3}}{2} \cdot \frac{1}{3} \right) - \frac{\frac{3}{3} \cdot \frac{1}{3}}{2} - \frac{\frac{1}{3} + \frac{2}{2}}{2} \cdot \frac{1}{3} - \frac{\frac{1}{3} \cdot \frac{1}{3}}{2} - \frac{\frac{2}{3} \cdot \frac{2}{3}}{2} - \frac{\frac{2}{3} \cdot \frac{1}{3}}{2} - \frac{\frac{1}{3} + \frac{2}{3}}{2} \cdot \frac{1}{3} - \left( \frac{1}{3} \right)^2 = \frac{32}{18} .$$

## 4 Generalizations of Pico formula in case of 3 - dimensional polyhedron

Suppose $P$ is a 3 - dimensional polyhedron without holes which contains $k+1$ vertexes with rational coordinates. As in the case of polygons, on the basis of non-descending ranges of coordinates

$$x_1, \quad x_2, \quad \ldots, \quad x_k, \quad x_{k+1}$$

$$y_1, \quad y_2, \quad \ldots, \quad y_k, \quad y_{k+1}$$

$$z_1, \quad z_2, \quad \ldots, \quad z_k, \quad z_{k+1}$$

we form the multitude $D = \{d_1^x, \quad d_2^x, \quad \ldots, \quad d_k^x, \quad d_1^y, \quad d_2^y, \quad \ldots, \quad d_k^y \quad d_1^z, \quad d_2^z, \quad \ldots, \quad d_k^z\}$ where

$$d_i^x = x_{i+1} - x_i, \quad \forall i = \overline{1,k} \ ,$$

$$d_i^y = y_{i+1} - y_i, \quad \forall i = \overline{1,k} \ ,$$

$$d_i^z = z_{i+1} - z_i, \quad \forall i = \overline{1,k} \ .$$

We'll denote by $d^{'}$ some of common divisors of the elements from multitude $D$. (We mention that for any multitude of rational numbers there exists, at least, one common divisor). In the space $\mathbb{R}^3$ we pass planes parallel to planes $XOY$, $XOZ$ and $YOZ$ so that the distance between any two parallel and neighbouring planes is equal to $d^{'}$. Thus in space $\mathbb{R}^3$ we obtain a cubic network which we'll denote by $Q(d^{'})$. In this case we can say that $P$ is a $d^{'}$-rational polyhedron. In a sequel we'll study the problem of volume calculation of polyhedron $P$ when this is a pyramid or prism.

**Definition 3.1** *One polyhedron $P$, the vertexes of which are situated in the knots of network $Q(d^{'})$, is named $d^{'}$-rational elementary polyhedron, if with exception of knots in which the vertexes of $P$ are situated, this contains no other knots from $Q(d^{'})$.*

Easily can be observed that if $P$ is a 1-rational elementary tetrahedron with height $h = 1$, then the volume of this is $V = \frac{1}{6}$. Surely, on the basis of those exposed earlier we obtain

$$V = \frac{1}{3} \cdot S_b \cdot h = \frac{1}{3} \cdot \left(i + \frac{b}{2} - 1\right) \cdot h = \frac{1}{3} \cdot \left(0 + \frac{3}{2} - 1\right) \cdot 1 = \frac{1}{6} \ .$$

In the case of $d^{'}$-rational tetrahedron with the height $H$ we have

$$V = \frac{(d^{'})^2 \cdot h}{6} \ .$$

**Theorem 3.1** *Any straight prism, the vertexes of which are situated in the knots of rational network $Q(d^{'})$, can be divided in $d^{'}$-rational elementary prisms.*

**Proof:** Firstly we observe that any polygon from plane, the vertexes of which are situated in the knots of rational network $D(d^{'})$ can be divided into $d^{'}$-rational elementary triangles, that is triangles which with exception of knots in which the vertexes of these are placed, contain no other knots of the network $D(d^{'})$. In this case we'll say that a triangulation of the polygon, determined by network $D(d^{'})$ is given.

One of possible triangulations of any polygon can iteratively be obtained in the following way:

1. We denote by $N(P)$ the multitude of knots of unitary network which belong to polygon $P$, that are situated on the frontier $bd(P)$ or in the interior $int(P)$ of this. Evidently $N(P) \neq \oslash$.

2. We choose an element $t \in N(P) \cap bd(P)$.

3. We form the multitude $\Gamma(t)$ of all $\omega \in N(P)$ knots, $\omega \neq t$, for which the segment $[t, \quad \omega]$ belongs to the polygon $P$, which contains no other knots of the network excepting $t$ and $\omega$, which can be united by a curve that belongs integrally to the polygon $P$.

4. We denote by $q^{'}$ and $q^{''}$ the elements from $\Gamma(t)$ which belong to the frontier $bd(P)$ and for which the curve line $[q^{'}, \quad t, \quad q^{''}]$ is placed on this frontier. We execute an order of elements of multitude $\Gamma(t)$ correspondingly to the clockwise direction, beginning with one of the knots $q^{'}$ or $q^{''}$, finishing with the second, with the condition that the curve which unites consecutive elements from $\Gamma(t)$ belongs as a whole to the polygon. Suppose $\Gamma(t) = \{s_1 = q^{'}, \quad s_2, \quad ..., \quad s_{k-1}, \quad s_k = q^{''}\}$ (see Figure 6 a) ). We draw up a curve, consequently uniting elements from $\Gamma(t)$. We denote by $P_\Gamma$ the polygon with the frontier $bd(P_\Gamma) = [s_1 = q^{'}, \quad s_2, \quad ..., \quad s_{k-1}, \quad s_k = q^{''}, \quad t, \quad q]$. We mention that $int(P_\Gamma)$ contains no knots of the network. Surely, in the opposite case such knot $z$ will belong to a triangle $[s_{i-1}, \quad t, \quad s_i]$, $i = \overline{2,k}$

which means that in the interior of the cone $s_{i-1}$, $t$, $s_1$ there exist nodes which weren't taken into consideration at forming the set $\Gamma(t)$. It is obvious that $P_\Gamma$ is a triangulated polygon.

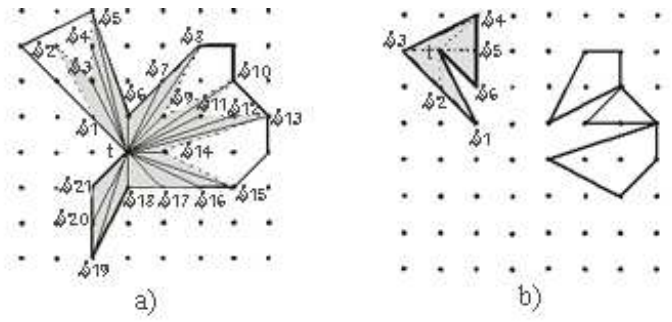


Figure 6.

5. Let us denote by $Bd(P)$ the common frontier between $P$ and $P_\Gamma$ and eliminate the set $Bd(P) \cap int(P_\Gamma)$ from $P$. If we obtain as a result a void set of the points from the plan then we obtain the triangulation of the initial polygon. Otherwise we denote by $P$ the remained part of the polygon. We mention that in general case the obtained domain $P$ can be a reunion of simple polygons. (see Figure 6 b). Let us return to the step 2 and continue the triangulation procedure.

If for the polygon $P$ from Figure 6 a) we consecutively apply the described algorithm, then we obtain the situations described in the Figure 6 a), b) and Figure 7 a), b).

Finally we'll obtain a triangulation of the initial polygon. Such a triangulation is presented in Figure 8. Of course, the triangulation of the polygon on the basis of the described algorithm is not obtained univocally. Surely, the number of the triangles into which the polygon will be divided is always the same.

Now let $P$ be an arbitrary prism, the vertexes of which are situated in the nodes of the network $Q(d')$. Let us triangulate the polygon

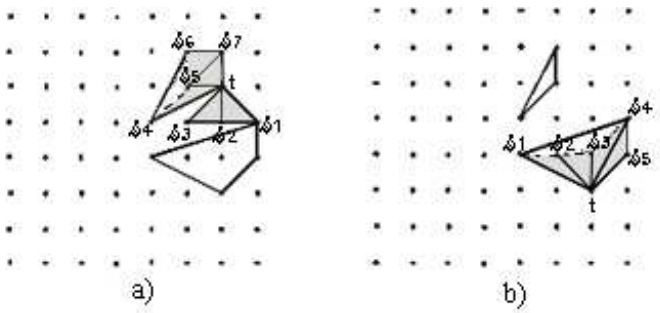


Figure 7.

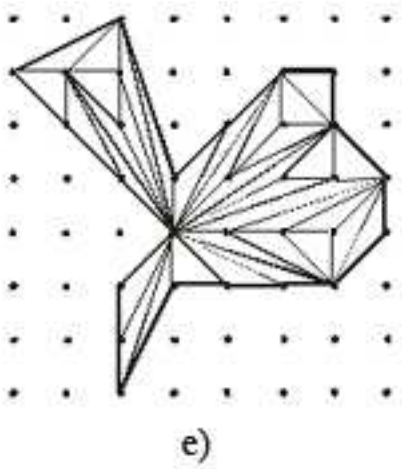


Figure 8.

from the base of the prism according to the described procedure. In accordance with the obtained base configuration we trace a vertical section of the prism $P$ and the planes parallel to the base through different nodes of the network. As a result we obtain a division of prism $P$ into $d^{'}$-rational elementary prisms.

**Consequence.** *Any prism the vertexes of which are situated in the nodes of the network $Q(d^{'})$ can be divided into $(2i+b-2)(n-1)$ $d^{'}$-rational elementary prisms, where $i$ is the number of nodes of the network situated in the interior of the polygon on the base of the prism, $b$ – the number of nodes on the polygon frontier, and $n$ – the number of nodes on a lateral edge.*

**Proof:** According to the Euler formula, the number of triangular domains into which a plane can be divided using $k$ points is $2k-4$. In the case of the studied polygon $k=i+b$. Because the domains from the polygon exterior do not interest us we obtain that the polygon can be divided into $2k-4-1-(b-3)=2(i+b)-4-1-b+3=2i+b-2$ triangles. Because there are $n$ nodes of the network on the lateral edge of the prism we trace $(n-1)$ sections parallel to the base and thus we obtain $(2i+b-2)(n-1)$ $d^{'}$-rational elementary prisms.

**Theorem 3.2.** *The volume of a prism with the vertexes in the nodes of a network $Q(d^{'})$ can be calculated by the formula:*

$$V=\frac{1}{2}\cdot(2i+b-2)\cdot(n-1)\cdot(d^{'})^{3}\ ,$$

*where $i$ is the number of nodes of the network $Q(d^{'})$ which is situated in the interior of the polygon on the prism base, $b$ – the number of nodes on the polygon frontier, and $n$ – the number of nodes on a lateral edge.*

**Proof:** It is known that a $d^{'}$-rational elementary prism can be divided into three $d^{'}$-rational elementary pyramids of equal volume. Thus, according to those mentioned above, the volume of such a prism is $\frac{1}{2}\cdot(d^{'})^{2}\cdot h=\frac{1}{2}\cdot(d^{'})^{3}$. Taking into consideration the consequence 3.1 we obtain the affirmation of the Theorem 3.2.

**Theorem 3.3.** *The volume of a pyramid, the vertexes of which are in the nodes of the network $Q(d')$ is calculated by the formula*

$$V = \frac{1}{6} \cdot (2i + b - 2) \cdot k \cdot (d')^3 \ ,$$

*where $i$ and $b$ represent the number of network nodes, that belong to the interior of the base of the pyramid and to the frontier of this base respectively, and $k$ – the number of nodes on the height traced from the pyramid vertex.*

**Proof:** According to those described above let us make a triangulation of pyramid base. Joining every node of the network that belong to the base with the vertex of pyramid, we obtain $(2i + b - 2)$ $d'$-rational elementary pyramids. Each of these pyramids has the height $h = d' \cdot (k - 1)$, where $k$ is the number of nodes that belong to the height traced from pyramid vertex. Thus for initial pyramid we have

$$V = (2i + b - 2) \cdot \frac{(d')^2 \cdot h}{6} = \frac{1}{6} \cdot (2i + b - 2) \cdot (k - 1) \cdot (d')^2 \ .$$

Theorem is proved.

## References

[1] Shashkin I., *The Euler characteristics*. Nauka, Moscow, 1984, pp.85 (in Russian).

S. Cataranciuc, M. Holban,

Received March 29, 2007

Moldova State University
E–mail: *caseg@usm.md*, *marinah82@mail.ru*

Computer Science Journal of Moldova, vol.**15**, no.1(43), 2007

# On Soundness for Time Workflow Nets

Inga Camerzan

**Abstract**

Workflow technology is widely used in order to offer companies a solution for managing business processes. Time management is a critical component of workflow management. In a workflow management system there is a delay between the moment an activity becomes enabled and the moment the activity is executed by a certain resource. The notion of correctness also called soundness for untimed workflow nets is extended for Time Workflow Nets and a characterisation of this property is given for two particular classes of Time Workflow Nets.

## 1 Introduction

Workflow technology has been introduced in order to model and manage business processes, but workflows have some disadvantages: they are inflexible, they do not support inter-operability, and the formal verification of their correctness is difficult. For solving these problems, workflows can be modeled using Petri Nets[6, 9], which are expressive, have a well defined semantic, a very accessible graphical representation and reach techniques for checking quantitative and qualitative properties.

A *workflow* represents the automatization of a complex process which consists of a set of interdependent activities, orientated towards the fulfilling of a certain objective. The applicability domains of workflows are: modeling, coordination, management of business processes. Workflows are based on *cases*, which are generated by external clients or they are generated internal. A case is an instance of a workflow. A

workflow process is designed to handle similar cases, specifying what action must be executed and in what order.

In this article we will define and use Time Workflow nets for modeling workflows, because Petri nets have well-known three advantages: simplicity, generality, and adaptability.

- *Simplicity* - a reduced number of elementary concepts, which can be combined in a large variety.
- *Generality* - the different kind of semantics (transition sequences, tracks, processes) are easy to associate with Petri nets.
- *Adaptability* - the modification of the basic model leads to special models which include different aspects like time, making it usable in different domains.

Workflow properties can be easily checked using the analysis techniques of Petri nets.

The correctness, effectiveness, and efficiency of the business processes supported by the workflow management system are vital to the organization. It is important to analyze a workflow process definition before it is put into production. In this article we will focus on *verification* (establishing the correctness of a workflow). For verification linear algebraic techniques and coverability graph analysis can be used. With these techniques it is known that such problem like boundness and liveness are decidable. That is why we will reduce soundness problem to boundness and liveness problems.

## 2 Workflows

As we mentioned above a set of interdependent atomic activities forms a workflow. Basic entities of a workflow are: *actions*, *agents* and activities *dependences*.

An action can take place if some preconditions are fulfilled and it yields some postconditions. For the execution of an action a *trigger*

is necessary; a trigger can be represented by the external conditions which lead to the execution of an enabled task.

We distinguish between four types of tasks: *automatic* - a task is triggered at the moment when it is enabled, *user* - a task is triggered by human participant, *message* - an external event (message) triggers en enabled task instance, *time* - an enabled task instance is triggered by a clock (we are especially interested in these types of tasks).

A task which is enabled for a specific case is a *work item*. A work item is the combination of action + case + trigger (optional).

An *activity* is the actual execution of a work item, i.e., a task is executed for a specific case = action + case + resourse (optional) + trigger.

A workflow has three dimensions: the case dimension, the process dimension, the resource dimension.

1. *Case dimension* specifying that every case is treated individually.
2. *Process dimension* specifying the workflow process, i.e., actions and routing for these actions.
3. *Resource dimension* specifying the what resources are grouped in roles and organizational units.

We will focus only on process dimension.

## 3 Time Workflow Nets

In this section we model the process dimension using Petri nets.

A Petri Net is a bipartite graph with two types of nodes: places and transitions interconnected by arcs, which connect only different types of nodes.

The process dimension specifies, as we mentioned above, which actions must be performed and in what order. For modeling workflows by means of Petri Nets the transition will be done directly: actions will be modeled by transitions, work items by enabled transitions, activities by firing transitions, conditions will be modeled by places, and cases will be modeled by tokens and dependences by arcs.

Further we will consider only Petri nets which describe the life cycle of one case. A Workflow net [2] will be defined as a Petri net which models the workflow process definition.

**Definition 3.1** *A Petri net $PN = (P, T, F, W)$ is a Workflow net iff:*

1. *PN has two additional places i and o, "start" place i, "destination" place o.*

2. *If we add a transition $t*$ to PN which connects o with i then the resulting Petri net is strongly connected.*

*(where $P$ - is a finite set of places, $T$ - is a finite set of transitions, $F \subseteq P \times T \cup T \times P$ - is the flow relation, $W : F \to N$ - is the weight function.)*

We define the *extended net* $PN' = (P', T', F')$ with $P' = P, T' = T \cup \{t*\}, F' = F \cup \{(o, t^*), (t^*, i)\}$ $W' = W \cup \{W(o, t*) = 1, W(t*, i) = 1\}$.

The notion of trigger defined in the paragraph above corresponds to an additional condition which must be fulfilled before the execution of the action, so it can be modeled by a token in an supplementary input location for the action.

There are different known methods of incorporating time in Petri nets: associating time delay to transition, associating time delay to places, associating time delay with arcs, associating time delays or time intervals to different types of objects of the net, associating stochastic time. Further we consider only Petri nets which have deterministic time associated to transitions, in the form of time intervals, defined by Merlin in 1972 [9] and then studied by Berthomeu-Menasche, Popova [10, 11, 12], Berthomieu-Diaz [4, 5], Boucheneb-Berthelot. Time Petri nets are classical Petri nets where for each transition $t$, a time interval $[a_t, b_t]$ is associated. The times $a_t, b_t$ are relative to the moment at which $t$ was last enabled. Assuming that $t$ was last enabled at time $c_t$, then $t$ may fire only after the time interval $[a_t + c_t, b_t + c_t]$ elapses.

We define a Time Workflow net in following way:

**Definition 3.2** *A Time Workflow net is a tuple* $\Sigma$*=(P, T, F, W, I) where PN=(P, T, F, W) is the Workflow net,* $I: T \to Q_0^+ \times Q_0^+$ *(where* $Q_0^+$ *is the set of assertive numbers) is a time function which associates timed intervals with transitions and for each transition* $t \in T$*,* $I_1(t) \leq I_2(t)$*, where* $I(t) = (I_1(t), I_2(t))$*.*

A global clock is associated with the Time Workflow net, which begins to work as soon as the first token appears in the net. After time association, the Workflow net will work in following way: from the moment when a transition $t$ is enabled, the tokens from the input locations are stored for $I_2(t) - I_1(t)$ time units, and after this time elapses the transition fires putting tokens in their output places. For transitions in conflict, the first transition that fires is the one which has the latest time interval smaller.

For the definition of a state and of a change of state of the net $\Sigma$ we will follow [10, 11]:

**Definition 3.3** *Let* $\Sigma = (P, T, F, W, I)$ *be a Time Workflow net and* $J : T \to Q_0^+ \cup \{\sharp\}$*. Then* $S = (m, J)$ *is the state of* $\Sigma$ *iff:*

1. $m$ *is a marking in skeleton net.*
2. $\forall t$ *(*$t \in T$ *and* $t^- \leq m \to J(t) \leq I_2(t)$*).*
3. $\forall t$ *(*$t \in T$ *and* $t^- \not\leq m \to J(t) = \sharp$*).*

*(where symbol* $\sharp$ *means that clock does not work,* $t^-(p) = W(p,t), \forall p \in P$ *is arc weight from place* $p$ *to transition* $t$*).*

One can understand the notion of state in the following way: let $S = (m, J)$ be a state. Each transition $t$ in the net has a watch. The watch doesn't work ($J(t) = \#$) at the marking $m$ if $t$ is disabled at $m$. If $t$ is enabled at $m$, then the watch of $t$ shows the time $J(t)$ that has elapsed since $t$ was last enabled.

Let $\Sigma = (P, T, F, W, I)$ be a time workflow net. The state $S_0 := (i, J_0)$ with $i$ the initial marking of the workflow net (the marking which

has a single token in place $i$) and $J_0(t) = \begin{cases} 0 & \text{iff} \quad t^- \leq m \\ \# & \text{iff} \quad t^- \not\leq m \end{cases}$ is considered to be the initial state of the time workflow net.

The states in a Time Worklow Net can change due to transition firings or time elapsing.

**Definition 3.4** *Transition $t$ is enabled in the state $S = (m, J)$, denoted by $S \rightarrow$ iff*

1. $t^- \leq m$
2. $I_1(t) \leq J(t)$.

The resulting state is defined as follows:

**Definition 3.5** *Transition $t$ enabled at the state $S = (m, J)$, will fire inducing state $S' = (m', J')$, denoted by $S \rightarrow S'$ defined thus:*

1. $m'(p) = m(p) + \triangle t(p) = m(p) + W(t,p) - W(p,t)$
2. $$J'(t) = \begin{cases} \sharp & t^- \not\leq m' \\ J(t) & t^- \leq m \wedge t^- \leq m\prime \wedge F_t \cap F'_t = 0 \\ 0, & otherwise \end{cases}$$

*where $F_t = \{p | p \in P \wedge pFt\}$, $F'_t = \{p | p \in P \wedge pF't\}$*

**Definition 3.6** *Let $\Sigma = (P, T, F, W, I)$ be a time workflow net. The state $S = (m, J)$ changes into the state $S' = (m', J')$ by the time duration $\tau \in Q$, denoted by $S \xrightarrow{\tau} S'$ iff: $m' = m$ and the time duration $\tau$ is possible $\forall t(t \in T \wedge J(t) \neq \# \rightarrow J(t) + \tau \leq I_2(t))$ and*
$J'(t) = \begin{cases} J(t) + \tau & iff \quad t^- \leq m \\ \# & iff \quad t^- \not\leq m \end{cases}$

**Definition 3.7** *$RS(\Sigma, S_0)$ denotes the set of all reachable states from initial state $S_0$.*

**Definition 3.8** *Transition $t$ is live at the state $S'$ iff $\forall S' \in RS(\Sigma, S_0) \rightarrow \exists S''(S'' \in RS(\Sigma, S')$ and $t$ is enabled at the state $S'')$. State $S$ is live in the net $\Sigma$ iff all transitions $t \in T$ are live in $S$ and $\Sigma$ is live iff $S_0$ is live in $\Sigma$.*

**Definition 3.9** *The state $S$ is bounded iff $\forall p \in P : \exists k \in N : \forall S' \in [S > S(p) \leq k$. The net $\Sigma$ is bounded if $\forall S \in [S_0 >$ is bounded.*

# 4 The Soundness Property

This section defines a notion of correctness for Time Workflow Nets - the notion of soundness and a sufficient condition for soundness is proven. This property reduces to the problem of soundness for the corresponding untimed workflow skeleton net, for two special classes of Time Workflow Nets: time interval workflow nets with immediate transitions: for these nets, a transition can fire as soon as it becomes enabled, the second class is the class of time interval workflow nets with transitions that are not forced to fire in a specific amount of time. For these classes, the soundness property can be checked by verifying the boundness and liveness property for an untimed Petri Net.

**Definition 4.1** *Let $\Sigma = (P, T, F, W, I)$ be a Time Workflow Net. $\Sigma$ is sound iff:*

1. *For every state $S$ reachable from the initial state $S_0$, there exists a firing sequence leading from $S$ to a final state $(o, J)$*
   $\forall S(S_0[*\rangle S) \Rightarrow (S[*\rangle(o, J)$
2. *The states $(o, J)$ are the only states reachable from state $S_0$ with at least one token in place $o$:*
   $\forall S = (M, J)(S_0[*\rangle S \wedge M \geq o \Rightarrow (M = o)$
3. *There are no dead transitions in $\Sigma$:*
   $\forall t \in T, \exists S, S'(S_0[*\rangle S[t\rangle S')$

Note that the soundness property relates to the dynamics of the WF - net. Given $\Sigma = (P, T, F, W, I)$ a Time Workflow Net, we define the

extended Time Workflow Net $\Sigma'$ as follows: $\Sigma' = (P', T', F', W', I')$ where:

* $P' = P$
* $T' = T \cup \{t^*\}$
* $F' = F \cup \{(o, t^*), (t^*, i)\}$
* $I'(t) = I(t)$ for all $t \in T$ and $I'(t^*) = [0, +\infty]$
* $W' = W \cup \{W(o, t*) = 1, W(t*, i) = 1\}$

**Lemma 4.1** *Let $\Sigma$ be a time workflow net with the initial state $S_0 = (i, J_0)$. If $\Sigma'$ is live and bounded, then $\Sigma$ is a sound time workflow net.*

**Proof.**
$\Sigma'$ is live, i.e for each reachable state $S$ there is a sequence which leads to another state $S'$ in which transition $t^*$ is enabled. Let $S' = (m', J')$. Since $t^*$ can fire it results that $t^*$ is enabled in marking $m'$. Place $o$ is the input place for $t^*$, so $m'(o) = 1$. So, for any state reachable from the initial state, it is possible to reach a state with at least one token in place $o$. So the first condition from the definition of soundness holds.

Consider $S$ a state reachable from $S_0$, $S = (M, J)$ with $M \geq o$ (at least one token in place $o$). This means $M = M' + o$. The transition $t^*$ is fireable in this marking: Since $M \geq o$, then $M(o) \geq 1$ and $o$ is the only input place for $t^*$, so $t^*$ is enabled in $M'$. It also holds that $I_1(t^*) \leq J(t^*)$ because $I_1(t^*) = 0$. If $t^*$ fires, a new state $S' = (M' + i, J')$ is reached. Since $\Sigma'$ is bounded and $M' + i \geq i$ it results that $M' + i = i$, so $M'$ should be equal to the empty state. Hence condition (2) from the definition of soundness also holds. The final condition from the definition of soundness results from the fact that $\Sigma'$ is live.

The next lemma shows that, for time interval workflows nets with immediate transitions (i.e transitions that can fire as soon as they become enabled), the soundness property implies the boundness of the extended time workflow net.

**Lemma 4.2** *If $\Sigma$ is sound and $\forall t \in T : I_1(t) = 0$ then $\Sigma'$ is bounded.*

**Proof.**
We will first show that $\Sigma$ is bounded. Assume that $\Sigma$ is sound and $\Sigma$ is not bounded. Since $\Sigma$ is not bounded, there are two states $S_i = (M_i, J_i)$ and $S_k = (M_k, J_k)$ such that $S_0[*\rangle S_i, S_i[*\rangle S_k$ and $M_k > M_i$. However, since $\Sigma$ is sound, there is a sequence $\sigma = \tau_0, t_0, \dots \tau_{n-1}, t_{n-1}$ such that $S_i[\sigma\rangle(o, J)$. We will show that the sequence $\sigma' = t_0, t_1, \dots t_{n-1}$ is fireable from $S_k$. We prove the statement by induction on $n$.

If $n = 1$, then $S_i \stackrel{\tau_0}{\to} S'_{i1} \stackrel{t_0}{\to} S_{i1}$. We prove that $t_0$ is fireable at state $S_k$. It holds that $t_0^- \leq M_k$, since $t_0^- \leq M_i$ and $M_k > M_i$. It must hold that $J_k(t_0) \geq I_1(t_0)$. But this always holds, because $I_1(t_0) = 0$. It also holds that $M_{k1} > M_{i1}$.

Suppose the statement holds for $n$ and we want to prove it for $n+1$. So, if $S_i \stackrel{\tau_0}{\to} S'_{i1} \stackrel{t_0}{\to} S_{i1}, \dots \stackrel{\tau_{n-1}}{\to} S'_{in} \stackrel{t_{n-1}}{\to} S_{in}$ then we have the sequence: $S_k \stackrel{t_0}{\to} S_{k1} \dots S_{kn-1} \stackrel{t_{n-1}}{\to} S_{kn}$. Let $S_i \stackrel{\tau_0}{\to} S'_{i1} \stackrel{t_0}{\to} S_{i1}, \dots \stackrel{\tau_{n-1}}{\to} S'_{in} \stackrel{t_{n-1}}{\to} S_{in} \stackrel{\tau_n}{\to} S'_{in+1} \stackrel{t_n}{\to} S_{in+1}$. From the induction assumption: $S_k \stackrel{t_0}{\to} S_{k1} \dots S_{kn-1} \stackrel{t_{n-1}}{\to} S_{kn}$. It also holds that: $M'_{ij+1} = M_{ij} < M_{kj}, j \in 0 \dots n-1$. We prove that $t_n$ is fireable at state $S_{kn}$ and $M_{kn+1} > M_{in+1}$. We know that $t_n$ is fireable at $S'_{in+1}$ and $M'_{in+1} = M_{in} < M_{kn}$ hence $t_n^- \leq M_{kn}$. It must hold that $J_{kn}(t_n) \geq I_1(t_n)$. This statement always holds, since $I_1(t_n) = 0$. So $t_n$ can fire at state $S_{kn}$ and it results a new state $S_{kn+1}$ with $M_{kn+1} > M_{in+1}$.

Using the result proven above, if $S_i[\sigma\rangle S_o$ and $S_i > S_k$ then there exists $\sigma'$ such that $S_k[\sigma'\rangle S_{ko}$ such that $M_{ko} > o$. This fact contradicts the condition 2 from the definition of soundness. Thus, $\Sigma$ must be bounded. From the fact that $\Sigma$ is bounded and sound it results that $\Sigma'$ is bounded: if transition $t^*$ in $\Sigma'$ fires, then the time workflow net returns to its initial state.

The next lemma proves the same result as Lemma 4.2 for time interval workflow nets in which transitions don't have the obligation to fire at a specific moment of time.

**Lemma 4.3** *If $\Sigma$ is sound and $\forall t \in T : I_2(t) = +\infty$ then $\Sigma'$ is bounded.*

**Proof.**
We will first show that $\Sigma$ is bounded. Assume that $\Sigma$ is sound and $\Sigma$ is not bounded. Since $\Sigma$ is not bounded, there are two states $S_i = (M_i, J_i)$ and $S_k = (M_k, J_k)$ such that $S_0[*\rangle S_i, S_i[*\rangle S_k$ and $M_k > M_i$.

We will prove that for any sequence $\sigma = S_i \stackrel{\tau_0}{\rightarrow} S'_{i1} \stackrel{t_0}{\rightarrow} S_{i1}, \ldots \stackrel{\tau_{n-1}}{\rightarrow} S'_{in} \stackrel{t_{n-1}}{\rightarrow} S_{in}$ there exists a sequence $\sigma'$ fireable from $S_k$: $\sigma' = S_k \stackrel{\tau_0^*}{\rightarrow} S'_{k1} \stackrel{t_0}{\rightarrow} S_{k1}, \ldots \stackrel{\tau_{n-1}^*}{\rightarrow} S'_{kn} \stackrel{t_{n-1}}{\rightarrow} S_{kn}$ with $M_{kn} > M_{in}$, where $\tau_i^* = max\{I_1(t), t^- \leq M_{i-1}\}$.

We will prove that if $S_{il-1} \stackrel{\tau_l}{\rightarrow} S'_{il} \stackrel{t_l}{\rightarrow} S_{il}, \quad M_{il-1} > M_{ik-1}$ then it holds:
$S_{kl-1} \stackrel{\tau_l^*}{\rightarrow} S'_{kl} \stackrel{t_l}{\rightarrow} S_{kl}$ and $M_{kl} > M_{il}$, where $\tau_l^* = max\{I_1(t), t^- \leq M_{il-1}\}$. We must prove that the time duration $\tau_l^*$ is possible at $S_{il-1}$, i.e $\forall t : t^- \leq M_{il-1} \rightarrow J_{il-1}(t) + \tau_l^* \leq I_2(t)$. This always holds, since $I_2(t) = +\infty$. The resulting state has $J'_{kl}(t) = J_{kl-1}(t) + \tau_l^*, \forall t : t^- \leq M_{il-1}$. Next, we must prove that $t_l$ is fireable at the state $S'_{kl}$. We know that $M'_{kl} = M_{kl-1} > M_{il-1}$ and $t_l$ is fireable at $M'_{il} = M_{il-1}$, so it holds that $t_l^- \leq M'_{kl} = M_{kl-1}$. We must prove that $I_1(t_l) \leq J'_{kl}(t_l)$. But $t_l^- \leq M_{kl-1} = M'_{kl}$, so, from the definition of $\tau_l^*$ it holds that $\tau_l^* \geq I_1(t_l)$. Then, $J'_{kl}(t_l) = J_{kl-1}(t_l) + \tau_l^* \geq J_{kl-1}(t_l) + I_1(t_l) \geq I_1(t_l)$. So, $I_1(t_l) \leq J'_{kl}(t_l)$. Thus we have proven that $t_l$ is fireable at $S'_{kl}$. For the resulting state $S_{kl}$ it holds that $M_{kl} > M_{il}$.

From the fact that $\Sigma$ is sound, it results that there exists a sequence $\sigma$ such that $M_i[\sigma\rangle(o, J) = M_o$. Then, there exists a sequence $\sigma'$ as described above such that $M_k[\sigma'\rangle M'_o = (M', J')$ such that $M' > o$. This relation contradicts the second relation from the definition of soundness, so $\Sigma$ cannot be unbounded. From the fact that $\Sigma$ is bounded and sound it results that $\Sigma'$ is bounded: if transition $t^*$ in $\Sigma'$ fires, then the time workflow net returns to its initial state.

**Lemma 4.4** *If $\Sigma$ is a sound time workflow net, then $\Sigma'$ is live.*

**Proof.**
First we show that state $S_0$ is a home state for $\Sigma'$, i.e $\forall S \in [S_0\rangle_{\Sigma'} : S_0 \in [S\rangle_{\Sigma'}$. From the definition of soundness, for all states $S \in [S_0\rangle$,

there exists an execution sequence $S[\sigma\rangle(o, J)$. We prove that $t^*$ is enabled in state $(o, J)$. It holds that $t^{*-} \leq o$. We must prove that $J(t^*) \geq I_1(t^*)$. This relation is true, because $I_1(t^*) = 0$. The resulting state is $S_0 = (i, J_0)$ So, for every state $S \in [S_0\rangle$ there exists a sequence $S[\sigma\rangle t^*[S_0\rangle$. Now we will prove that $\Sigma'$ is live. Let $t$ be a transition and $S$ a state. From the soundness (3), there exists state $S' \in [S_0\rangle_\Sigma$ such that $t$ is enabled in $S'$. We show that $S' \in [S\rangle_{\Sigma'}$. We know that $S[*\rangle_{\Sigma'} S_0[*\rangle_{\Sigma'} S'$. So $S' \in [S\rangle_{\Sigma'}$, and we have proven that $\Sigma'$ is live.

**Theorem 4.1** *If $\Sigma = (P, T, F, W, I)$ is a Time Workflow Net, such that $\forall t \in T : I_1(t) = 0$. Then $\Sigma$ is sound iff the extended Time Workflow Net, $\Sigma'$ is live and bounded.*

**Proof.**
The proof of the theorem results immediately from Lemma 4.1, Lemma 4.2 and Lemma 4.4.

**Theorem 4.2** *If $\Sigma = (P, T, F, W, I)$ is a Time Workflow Net, such that $\forall t \in T : I_2(t) = +\infty$. Then $\Sigma$ is sound iff the extended Time Workflow Net, $\Sigma'$ is live and bounded.*

**Proof.**
The proof of the theorem results immediately from Lemma 4.1, Lemma 4.3 and Lemma 4.4

**Proposition 4.1** *Let $\Sigma$=(P, T, F, W, I) be a Time Workflow Net such that $\forall t$ ($t \in T \rightarrow I_1(t) = 0$) and $S(\Sigma)$ the skeleton of $\Sigma$, then it holds:*

1. *$S(\Sigma)$ is unbounded iff $\Sigma$ is unbounded.*
2. *$S(\Sigma)$ is live iff $\Sigma$ is live.*

**Proof.**
Demonstration is similar to the demonstration from [11].

**Proposition 4.2** *Let $\Sigma$=(P, T, F, W, I) be a Time Workflow Net such that $\forall t(t \in T \rightarrow I_2(t) = \infty)$ and $S(\Sigma)$ the skeleton of $\Sigma$, then it holds:*

1. *$S(\Sigma)$ is unbounded iff $\Sigma$ is unbounded.*

2. *$S(\Sigma)$ is live iff $\Sigma$ is live.*

It can be noticed that the two classes of Time Workflow Nets defined above have the same boundedness and liveness behaviour as the corresponding classical Petri Nets (their skeletons). Now, using theorems 4.1 and 4.2 and proposition 4.1 and 4.2, the following results regarding the soundness of these classes of Time Workflow Nets can be proven.

**Theorem 4.3** *Let $\Sigma = (P, T, F, W, I)$ be a Time Workflow Net such that $\forall t$ $(t \in T \rightarrow I_1(t) = 0)$ and $S(\Sigma)$ the skeleton of $\Sigma$, then it holds: $\Sigma$ is a sound Time Workflow Net iff $S(\Sigma)$ is a sound workflow net.*

**Proof.**
According to Theorem 4.1, $\Sigma$ is sound iff $\Sigma'$ is live and bound. Since $\Sigma'$ has for all $t : I_1(t) = 0$, then $\Sigma'$ is live and bound iff $S(\Sigma')$ is live and bound. But $S(\Sigma') = S(\Sigma)'$, so $S(\Sigma')$ is live and bound iff $S(\Sigma)'$ is live and bound. For untimed workflow nets we know that $WF$ is sound iff $WF'$ is live and bounded. So $S(\Sigma)$ is sound.

**Theorem 4.4** *Let $\Sigma$=(P, T, F, W, I) be a Time Workflow Net such that $\forall t$ $(t \in T \rightarrow I_2(t) = \infty)$ and $S(\Sigma)$ the skeleton of $\Sigma$, then it holds: $\Sigma$ is a sound Time Workflow Net iff $S(\Sigma)$ is a sound workflow net.*

Using theorem 4.3, theorem 4.4 and the fact that soundness is decidable for untimed workflow nets, it results that:

**Corollary 4.1** *Let $\Sigma = (P, T, F, W, I)$ be a Time Workflow Net such that $\forall t$ $(t \in T \rightarrow I_1(t) = 0)$. The soundness property is decidable for $\Sigma$.*

**Corollary 4.2** *Let $\Sigma = (P, T, F, W, I)$ be a Time Workflow Net such that $\forall t$ $(t \in T \rightarrow I_2(t) = \infty)$. The soundness property is decidable for $\Sigma$.*

For the two classes of Time Workflow Nets described above, the soundness property is decidable and it can be checked by verifying the boundness and liveness property of the underlying untimed net of the extended net $\Sigma'$.

## 5 Conclusions

In this paper we have introduced a new class of Petri Nets for modelling workflows with time delays associated to tasks. We have defined the notion of soundness for Time Workflow Nets, extending the notion of soundness defined in [2] for untimed workflow nets. It was shown for a Time Workflow Net $\Sigma$ that, if the extended Time Workflow Net $\Sigma'$ is live and bounded, then $\Sigma$ is sound. There were identified two subclasses of Time Workflow Nets (Time Workflow Nets with immediate transition firing and Time Workflow Nets with no obligation to fire for transitions) for which the soundness property reduces to the soundness property of the skeleton net. Thus, the soundness can be verified using the liveness and the boundness properties of an untimed workflow net. Therefore, the soundness property is decidable in these two particular cases. Further we research aims at finding a characterisation for the soundness property for all Time Workflow Nets and finding interesting subclasses of Time Workflow Nets for which the soundness is decidable.

## References

[1] W. M. P. van der Aalst. *Structural Characterization of Sound Workflows nets*, Computing Science Reports 96-23, Eindhoven University of Technology, Eindhoven, 1996.

[2] W. M. P. van der Aalst. *The Application of Petri nets to Workflow Management*, The journal of Circuits, Systems and Computers, 8(1) : 21–66, Eindhoven University of Technology, The Netherlands, 1998.

[3] W. M. P. van der Aalst. *Verifications of Workflow Nets*, In P. Azema and G. Balbo, editors, Application and Theory of Petri

nets 1997, volume 1248 of Lecture Notes in Computer Science, pages 407 – 426, Springer - Verlag, Berlin, 1997.

[4] B. Berthomieu. *Modeling and Verification of Time Dependent Systems Using Time Petri Nets*, In Advances of petri Nets 1984, vol 17 , No 3 of IEEE Trans. On Software Eng. 1991, 259–273.

[5] B. Berthomieu. *An Enumerative Approach for Analyzing Time*, In Proceedings IFIP 1983, R:E:A:Mason(ed), North-Holland, 1983, 41–47.

[6] T. Jucan, F. Tiplea. *Retele Petri - teorie si practica*, Academia Romana, Bucuresti, 1999.

[7] J. Eder. *Time Constraints in Workflow Systems*, March 15, 1999.

[8] Li Hui - Fang. *Workflow Model Analysis Based on Time Constraint Petri Nets*, Journal of Software, vol 15, No.1,2004, 17–26.

[9] P. Merlin. *A study of the recoverability of computer system*, Ph. D. thesis, Dep. Computing Science, University California, Irvine, 1974.

[10] L. Popova-Zeugmann. *On Time Invariance in Time Petri Nets*, In Informatik-Bericht Nr. 36 der Institute pur Informatik der Humboldt-Univ. Zu Berlin, Oct. 1994

[11] L. Popova-Zeugmann. *On Liveness and Boundness in Time Petri Nets*

[12] L. Popova-Zeugmann. *On Parametrical Sequences in Time Petri Nets*, Proceedings of the CSP 97 Workshop, Warsaw(1997), 105–111.

[13] WFMC. *Workflow Management Coaliton Terminology and Glosary (WFMC-TC-1011)*, Technical Report, The Workflow Management Coalition, Brussels 1999.

Inga Camerzan,

State University of Tiraspol
E–mail: *caminga2002@yahoo.com*

Received January 31, 2007

Computer Science Journal of Moldova, vol.**15**, no.1(43), 2007

# Ordering of jobs with three different processing times in the $Mxn$ Bellman-Johnson problem

Ion Bolun

**Abstract**

Bellman-Johnson *Mxn* scheduling problem with monotone (no decreasing, constant or no increasing) jobs of three different processing times is investigated. Three different classes $C_{3.1}$, $C_{3.2}$ and $C_{3.3}$ of such systems are considered. On the basis of earlier results, the solution for optimal ordering of adjacent or nonadjacent jobs in pairs for each of these classes of systems is obtained. In addition, examples of systems for which it is possible to obtain the optimal solution of ordering all $n$ jobs are done, too.

## 1 Introduction

Bellman-Johnson *Mxn* scheduling problem in sequential systems [1] – one of the main problem in theory of scheduling [2, 3], is not solved, yet. Solutions for some particular cases only are obtained [1-5, 7, 8] and algorithms for quasi optimal solving of the general problem are proposed [5, 6]. The notion of *monotone jobs* is defined in [7]. There, some results referring to partial or total ordering of no decreasing, constant or no increasing jobs are obtained, too. In article [8], the case of monotone jobs with no more than two different processing times is investigated.

In this paper, some particular cases of partial or total ordering of jobs with no more than three different processing times are investigated. For each such a job, the processing time on first sequence of processing units (servers) is the same, on the second sequence of servers is the same too, although possible different from the first one, and on the

third sequence of servers is also the same, although possible different from the first two ones.

## 2 Preliminary considerations

The *Mxn* Bellman-Johnson problem foresees the execution of $n$ jobs by a system of $M$ consecutive servers. Each server processes, at any moment of time, only one job and may begin the execution of next job immediately after completion of the current one. Jobs' processing order must be the same on all systems' servers. It is required to determine the order, which assure the minimal total processing time $T$ of the $n$ jobs:

$$T = \max_{1 \le u_1 \le u_2 \le \ldots \le u_{m-1} \le n} \left( \sum_{j=1}^{M} \sum_{k=u_{j-1}}^{u_j} \tau_{j i_k} \right) \to \min, \tag{1}$$

where $\tau_{j i_k}$ is the processing time on server $j$ of job $i_k$, placed in the schedule on place $k$, and, also, $u_0 = 1$, $u_M = n$.

Let $\Omega = \{1, 2, 3, ..., n\}$ be the set of all jobs to be processed in the system. From earlier known results referring to jobs ordering, below we address, in particular, to Statement 5 and Consequence 4 from paper [5] and to Statements 2, 3, to Consequence 1, to Statements 4, 8, 5 and 9 from paper [7], which in this paper are described as Statements 1-9, respectively, but without their proof.

**Statement 1** [5]. Let, for a pair of jobs $\alpha$ and $\beta$ from the $n$ ones, the following relations take place

$$\min(\tau_{j\,\alpha}; \tau_{j+1,\,\beta}) \le \min(\tau_{j+1,\,\alpha}; \tau_{j\,\beta}), j = \overline{1, M-1} \tag{2}$$

and, at the same time, let for a server $v \in [2, M]$ the inequality

$$\tau_{v\alpha} < \tau_{v\beta} \tag{3}$$

takes place and for a server $k \in [2, v-1]$ the equality

$$\tau_{k\alpha} = \tau_{k\beta} \tag{4}$$

takes place; in these conditions, if the inequality

$$\tau_{k\alpha} \geq \tau_{k-1,\alpha} \tag{5}$$

takes place too, then when placing jobs $\alpha$ and $\beta$ near each other in the schedule it is opportune, in sense of (1), that $\alpha \to \beta$ (job $\alpha$ precedes to job $\beta$).

**Statement 2** [5]. If, for any pair $\alpha$, $\beta$ from the $n$ jobs, relations (2)-(5) take place and these are transitive ones, then the optimal, in sense of (1), schedule can be obtained according to the rule: $\alpha \to \beta$, if conditions (2)-(5) are satisfied.

**Statement 3** [1, 7]. Conditions (2) are transitive ones, in other words, if relations (2) and relations $\min(\tau_{j\beta}; \tau_{j+1,\gamma}) \leq \min(\tau_{j+1,\beta}; \tau_{j\gamma})$, $j = \overline{1, M-1}$ take place, then relations $\min(\tau_{j\alpha}; \tau_{j+1,\gamma}) \leq \min(\tau_{j+1,\alpha}; \tau_{j\gamma})$, $j = \overline{1, M-1}$ take place, too.

**Statement 4** [7]. At $\alpha \in A$, the conditions (3)-(5) are satisfied.

**Statement 5** [7]. At $\alpha \in A$, the conditions (2)-(5) are transitive ones.

**Statement 6** [7]. If relations

$$\tau_{j\alpha} = \tau_{j+1,\alpha}, \tau_{j\beta} = \tau_{j+1,\beta}, j = \overline{s, u} \tag{6}$$

take place, then the subset of conditions (2) for jobs $\alpha$ and $\beta$ on the server fragment $[s; u]$ is satisfied.

**Statement 7** [7]. When placing jobs $\alpha \in A$ and $\beta \in E$ near each other in the schedule, it is opportune, in sense of (1), that $\alpha \to \beta$.

**Statement 8** [7]. Let $L = \overline{i_l, i_{l+r-1}}$ and $L \subseteq C$, then it is unimportant, in sense of (1), the reciprocal placement of subset's $L$ jobs in the schedule on places $\overline{l, l+r-1}$ – this can be an arbitrary one.

**Statement 9** [7]. Let $L = \overline{i_l, i_{l+r-1}}$ and $L \subseteq (A \cup E) \subseteq \Omega$, then the rearrangement in the schedule of different categories of subsets of jobs from $L$ on places $\overline{l, l+r-1}$ is opportune, in sense of (1), according to the order: 1)$L \cap (A \backslash C) \to L \cap C \to L \cap (E \backslash C)$ or 2) $L \cap A \to L \cap (E \backslash C)$ or 3) $L \cap (A \backslash C) \to L \cap E$.

Below, the following definitions referring to monotone jobs, proposed in paper [7], and the definition regarding monotone jobs with three different processing times are used:

1. *No decreasing jobs* are those from the $n$ ones, for which relations

$$\tau_{ji} \leq \tau_{j+1,i},\ i \in A,\ j = \overline{1, M-1} \tag{7}$$

take place. Here $A$ is the set of all no decreasing jobs from the $n$ ones.

2. *No increasing jobs* are those from the $n$ ones, for which relations

$$\tau_{ji} \geq \tau_{j+1,i},\ i \in E,\ j = \overline{1, M-1}, \tag{8}$$

take place. Here $E$ is the set of all no increasing jobs from the $n$ ones.

3. *Constants jobs* are those from the $n$ ones, for which relations

$$\tau_{ji} = \tau_i,\ i \in C,\ j = \overline{1, M}, \tag{9}$$

take place. Here $C = C_1$ is the set of all constant jobs from the $n$ ones. From relations (7) – (9), one can see that

$$C = C_1 \subseteq (A \cup E). \tag{10}$$

4. *Monotone with three different processing times jobs* (of type $C_3$) are those from the $n$ ones, for which relations

$$\tau_{ji} = \begin{cases} \tau_{1i}, j = \overline{1, \nu_i} \\ \theta_i, j = \overline{\nu_i + 1, k_i} \\ \tau_{Mi}, j = \overline{k_i + 1, M} \end{cases}, i \in C_3, \tag{11}$$

take place. Here $C_3$ is the set of all monotone jobs with three different processing times from the $n$ ones. The processing time of job $i$ on first sequence of servers, namely $j = \overline{1, \nu_i}$, is $\tau_{1i}$, on second sequence of servers, namely $j = \overline{\nu_i + 1, k_i}$, is $\theta_i$, and on third sequence of servers, namely $j = \overline{k_i + 1, M}$, is $\tau_{Mi}$. From relations (7), (8) and (11), one can easy observe that relation

$$C_3 \subseteq (A \cup E) \tag{12}$$

takes place.

From the multitude of possible particular cases, referring to the set $C_3$ of jobs, the following three cases are investigated:

1) $C_3 = C_{3.1}$, where the set $C_{3.1}$ is constituted from $n$ jobs of $C_3$ type for which the equalities $\tau_{1i} = \tau$, $i = \overline{1,n}$ take place;

2) $C_3 = C_{3.2}$, where the set $C_{3.2}$ is constituted from $n$ jobs of $C_3$ type for which the equalities $\theta_i = \tau$, $i = \overline{1,n}$ take place;

3) $C_3 = C_{3.3}$, where the set $C_{3.3}$ is constituted from $n$ jobs of $C_3$ type for which the equalities $\tau_{Mi} = \tau$, $i = \overline{1,n}$ take place.

The proof of statements, with regard to jobs of types $C_{3.1}$, $C_{3.2}$ and $C_{3.3}$ ordering formulated below, is done by confirming the satisfaction of conditions (2)-(5) from Statement 1, for jobs placed near each other in the schedule, or of those of Statement 2, for general ordering of jobs in the schedule. According to their description, conditions (3)-(5) are satisfied at that time, when for the definition domain, outlined by relations (3) and (4), the relation (5) takes place; if this definition domain is empty, then it is not needed to satisfy relation (5) and is considered that conditions (3)-(5) are satisfied.

## 3 Ordering of $C_{3.2}$ type jobs

Let us consider a particular set $C_{3.2}$ of no decreasing (of type $A$) or no increasing (of type $E$) $n$ jobs with the following processing times:

$$\tau_{ji} = \begin{cases} \tau_{1i}, j = \overline{1,\nu_i} \\ \tau, j = \overline{\nu_i + 1, \kappa_i} \\ \tau_{Mi}, j = \overline{\kappa_i + 1, M} \end{cases}, i = \overline{1,n}, \tag{13}$$

accepting, to extend the implicated categories of jobs, that there can be $\kappa_i = \nu_i$, too, when job $i$ is with only two different processing times ($\tau_{1i}$ and $\tau_{Mi}$). One example of two jobs $\alpha$ and $\beta$ of type $C_{3.2}$ is shown in Figure 1.

**Statement 10.** For the set of jobs defined by relations (13), it is opportune, in sense of (1), that $\alpha \to \beta$ if $\alpha \in A$ and $\beta \in E$ or if relations:

$$\min(\tau_{1\alpha}; \tau_{M\beta}) \leq \min(\tau_{M\alpha}; \tau_{1\beta}), \tag{14}$$

$$\nu_\alpha \geq \nu_\beta, \tag{15}$$

$$\kappa_\alpha \quad \geq \quad \kappa_\beta \tag{16}$$

take place.

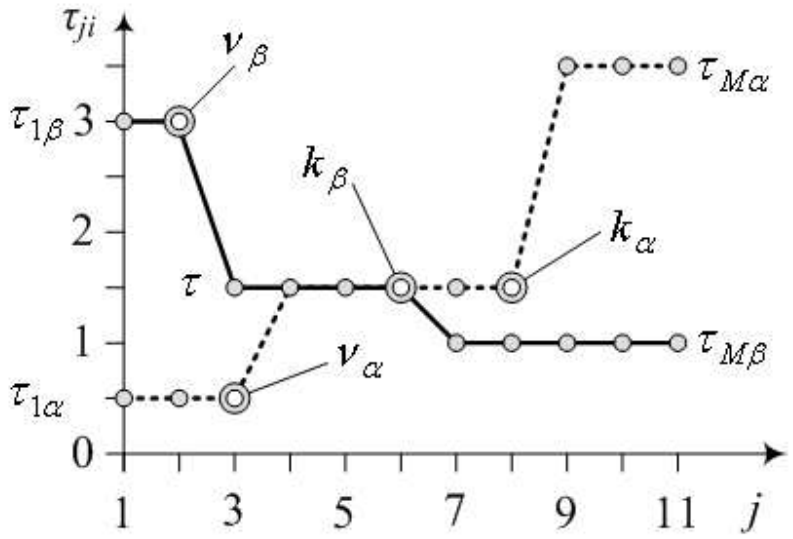


Figure 1. Two jobs $\alpha$ and $\beta$ of type $C_{3.2}$.

*Proof.* According to (12), the jobs from the set $C_3 = C_{3.2}$ are monotone no decreasing (belong to set $A$) or no increasing (belong to set $E$) ones. At the same time, on the basis of Statement 7, if $\alpha \in A$, $\beta \in E$ and $C_{3.2} = (A \cup E)$, then the placement of jobs $\alpha$ and $\beta$ in the schedule is opportune, in sense of (1), in order $\alpha \to \beta$. Hence, it remains to prove the reliability of this statement for cases $(\alpha, \beta) \in A$ and $(\alpha, \beta) \in E$. Proof will be done by confirming the satisfaction of conditions (2)-(5) from the Statement 1 and of their transitivity (according to Statement 2).

From relations (13), it is easy to see that $\nu_\alpha \leq \kappa_\alpha$ and $\nu_\beta \leq \kappa_\beta$. Therefore, for two concrete compared jobs $\alpha$ and $\beta$, there can be the following six variants of relations among values $\nu_\alpha$, $\kappa_\alpha$, $\nu_\beta$ and $\kappa_\beta$:

$$\left.\begin{array}{ll} 1)\ \nu_\alpha \le \kappa_\alpha \le \nu_\beta \le \kappa_\beta; & 2)\ \nu_\alpha \le \nu_\beta \le \kappa_\alpha \le \kappa_\beta; \\ 3)\ \nu_\alpha \le \nu_\beta \le \kappa_\beta \le \kappa_\alpha; & 4)\ \nu_\beta \le \nu_\alpha \le \kappa_\alpha \le \kappa_\beta; \\ 5)\ \nu_\beta \le \nu_\alpha \le \kappa_\beta \le \kappa_\alpha; & 6)\ \nu_\beta \le \kappa_\beta \le \nu_\alpha \le \kappa_\alpha. \end{array}\right\} \qquad (17)$$

The procedure for confirming the satisfaction of conditions (2) for each of the six variants (17) is the following. Let us consider the variant 1: $\nu_\alpha \le \kappa_\alpha \le \nu_\beta \le \kappa_\beta$. According to Statement 6, the conditions (2) on server fragments $[s; u]$, for which relations (6) take place, are satisfied. That's why, for variant 1 from (17), it is also necessary to verify the following 10 cases:

$$\left.\begin{array}{ll} 1.1)\ j = \nu_\alpha < \kappa_\alpha; & 1.2)\ j = \nu_\alpha = \kappa_\alpha < \nu_\beta; \\ 1.3)\ j = \nu_\alpha = \kappa_\alpha = \nu_\beta < \kappa_\beta; & 1.4)\ j = \nu_\alpha = \kappa_\alpha = \nu_\beta = \kappa_\beta; \\ 1.5)\ \nu_\alpha < j = \kappa_\alpha < \nu_\beta; & 1.6)\ \nu_\alpha < j = \kappa_\alpha = \nu_\beta < \kappa_\beta; \\ 1.7)\ \nu_\alpha < j = \kappa_\alpha = \nu_\beta = \kappa_\beta; & 1.8)\ \kappa_\alpha < j = \nu_\beta < \kappa_\beta; \\ 1.9)\ \kappa_\alpha < j = \nu_\beta = \kappa_\beta; & 1.10)\ \nu_\beta < j = \kappa_\beta. \end{array}\right\} \qquad (18)$$

For each of the ten cases from (18), it is needed to verify the respective condition from (2), taking into account relations (13) and (14). For example, for cases 1.5 and 1.7 from (18), one has, respectively:

$$1.5)\ \min(\tau; \tau_{1\beta}) \le \min(\tau_{M\alpha}; \tau_{1\beta}); \qquad (19)$$

$$1.7)\ \min(\tau; \tau_{M\beta}) \le \min(\tau_{M\alpha}; \tau_{1\beta}). \qquad (20)$$

If $(\alpha, \beta) \in A$ then, on the basis of relations (7), (8) and (13), inequalities $\tau_{1\beta} \le \tau \le \tau_{M\alpha}$ take place, hence condition (19) takes place, too. But the condition (20) doesn't take place, because according to (13) inequalities $\tau \ge \tau_{1\beta}$ and $\tau_{M\beta} \ge \tau_{1\beta}$ take place. In a similar mode, it was established that conditions (19) and (20) for cases 1.1, 1.2, 1.3, 1.4 and 1.5 at $(\alpha, \beta) \in A$ are satisfied.

If $(\alpha, \beta) \in E$ then, according to relations (7), (8) and (13), inequalities $\tau_{M\alpha} \le \tau \le \tau_{1\beta}$ take place, thus condition (19) doesn't take place. At the same time, because of $(\alpha, \beta) \in E$, relation $\tau_{M\beta} \le \tau_{1\beta}$ takes place and, according to relations (14), inequality $\tau_{M\beta} \le \tau_{M\alpha}$ takes place; hence condition (20) takes place, too. In a similar mode, it was

established that conditions (19) and (20) for cases 1.4, 1.7, 1.8, 1.9, 1.10 at $(\alpha, \beta) \in E$ are satisfied.

Combining cases $(\alpha, \beta) \in A$ and $(\alpha, \beta) \in E$, it is easy to obtain that conditions (2) for variant 1 take place only at

$$\nu_\alpha = \nu_\beta = \kappa_\alpha = \kappa_\beta, \tag{21}$$

which corresponds to the case with two different processing times for each job: $\tau_{1i}$ and $\tau_{Mi}$, $i = \overline{1,n}$.

Obtained results for the six variants (17) are specified in Table 1. In this table, the cases for each of variants 2-6 are formed in a similar mode as the formation of cases for variant 1, but taking into account the particular order, for the concrete variant, of values $\nu_\alpha$, $\kappa_\alpha$, $\nu_\beta$ and $\kappa_\beta$.

From Table 1 for each of the six variants, it is easy to observe that even if the set of cases, for which the local conditions are satisfied, is different (depending of job type), however the solution by job types is the same.

Table 1. Cases that satisfy conditions (2) for the set (13) of jobs

| Variant | Job type | Cases that satisfy local conditions (2) | Solution by job types | Solution for the variant |
|---|---|---|---|---|
| 1 | $(\alpha, \beta) \in A$ | 1.1-1.5 | $\nu_\alpha = \kappa_\alpha = \nu_\beta = \kappa_\beta$ | $\nu_\alpha = \kappa_\alpha = \nu_\beta = \kappa_\beta$ |
| | $(\alpha, \beta) \in E$ | 1.4, 1.7-1.10 | $\nu_\alpha = \kappa_\alpha = \nu_\beta = \kappa_\beta$ | |
| 2 | $(\alpha, \beta) \in A$ | 2.1-2.4, 2.8, 2.9 | $\nu_\alpha = \nu_\beta \le \kappa_\alpha = \kappa_\beta$ | $\nu_\alpha = \nu_\beta \le \kappa_\alpha = \kappa_\beta$ |
| | $(\alpha, \beta) \in E$ | 2.2, 2.4, 2.5, 2.7, 2.9, 2.10 | $\nu_\alpha = \nu_\beta \le \kappa_\alpha = \kappa_\beta$ | |
| 3 | $(\alpha, \beta) \in A$ | 3.1-3.4, 3.8-3.10 | $\nu_\alpha = \nu_\beta \le \kappa_\beta \le \kappa_\alpha$ | $\nu_\alpha = \nu_\beta \le \kappa_\beta \le \kappa_\alpha$ |
| | $(\alpha, \beta) \in E$ | 3.2-3.10 | $\nu_\alpha = \nu_\beta \le \kappa_\beta \le \kappa_\alpha$ | |
| 4 | $(\alpha, \beta) \in A$ | 4.1-4.9 | $\nu_\beta \le \nu_\alpha \le \kappa_\alpha = \kappa_\beta$ | $\nu_\beta \le \nu_\alpha \le \kappa_\alpha = \kappa_\beta$ |
| | $(\alpha, \beta) \in E$ | 4.1, 4.2, 4.4, 4.5, 4.7, 4.9-4.10 | $\nu_\beta \le \nu_\alpha \le \kappa_\alpha = \kappa_\beta$ | |
| 5 | $(\alpha, \beta) \in A$ | 5.1-5.10 | $\nu_\beta \le \nu_\alpha \le \kappa_\beta \le \kappa_\alpha$ | $\nu_\beta \le \nu_\alpha \le \kappa_\beta \le \kappa_\alpha$ |
| | $(\alpha, \beta) \in E$ | 5.1-5.10 | $\nu_\beta \le \nu_\alpha \le \kappa_\beta \le \kappa_\alpha$ | |
| 6 | $(\alpha, \beta) \in A$ | 6.1-6.10 | $\nu_\beta \le \kappa_\beta \le \nu_\alpha \le \kappa_\alpha$ | $\nu_\beta \le \kappa_\beta \le \nu_\alpha \le \kappa_\alpha$ |
| | $(\alpha, \beta) \in E$ | 6.1-6.10 | $\nu_\beta \le \kappa_\beta \le \nu_\alpha \le \kappa_\alpha$ | |

From the last column of Table 1 one can see: the solution for the

variant 1 is a particular case of solutions for variants 2-6; the solutions for variants 2-4 are particular cases for the variant 5 solution. Combining the solutions for variants 5 and 6 and taking into account that (see (13)) $\nu_\alpha \leq \kappa_\alpha$ and $\nu_\beta \leq \kappa_\beta$, one can obtain relations (15) and (16).

Thus, from conditions (2)-(5) of Statement 1, it remains to prove that conditions (3)-(5) take place. According to Statement 4, these ones take place at $\alpha \in A$. So, it is needed to prove that conditions (3)-(5) take place at $\alpha \in E$, too, that is at $(\alpha, \beta) \in E$, because case $\{\alpha \in E; \beta \in A\}$ signify that $\beta \to \alpha$ and therefore it secedes. Let $(\alpha, \beta) \in E$ and relations (13)-(16) take place. At $(\alpha, \beta) \in E$, the inequality (3) doesn't take place for server sequences:

- $j = \overline{\kappa_\alpha + 1, M}$, because according to relations (14) the inequality $\tau_{M\alpha} \geq \tau_{M\beta}$ takes place;
- $j = \overline{\kappa_\beta + 1, \kappa_\alpha}$, because according to relations (8) and (13) the inequality $\tau > \tau_{M\beta}$ takes place;
- $j = \overline{\nu_\alpha + 1, \kappa_\beta}$, because according to relations (13) the equalities $\tau_{j\alpha} = \tau_{j\beta} = \tau$ take place;
- $j = \overline{\nu_\beta + 1, \nu_\alpha}$, because according to relations (8) and (13) the relations $\tau_{j\alpha} = \tau_{1\alpha} \geq \tau = \tau_{j\beta}$ take place.

Thus inequality (3) can take place only at $j = \overline{1, \nu_\beta}$, but according to (13) in this case the inequality (4) doesn't take place. So, for $j = \overline{1, M}$, inequalities (3) and (4) don't take place concomitantly and the necessity of satisfaction the condition (5) secedes. Hence conditions (3)-(5), and with them *conditions (2)-(5), too, are satisfied.*

It remains *to prove the transitivity of conditions* (2)-(5). In this aim, it is sufficient to prove the transitivity of conditions (14)-(16), because, as was confirmed above in this section, if conditions (14)-(16) take place, then conditions (2)-(5) take place, too. According to Statement 3, conditions (2) are transitive ones, and conditions (14) are a particular case of conditions (2), so they are transitive, too.

One can easyly observe that conditions (15) and (16) are transitive, too. Really, if relations $\nu_\alpha \geq \nu_\beta$ and $\nu_\beta \geq \nu_\lambda$ take place, then the

inequality $\nu_\alpha \geq \nu_\gamma$ takes place, too. In a similar mode, if relations $\kappa_\alpha \geq \kappa_\beta$ and $\kappa_\beta \geq \kappa_\gamma$ take place, then the inequality $k_\alpha \geq k_\gamma$ takes place, too. Hence *conditions (14)-(16) are transitive ones*, that was required to be proved.

**Statement 11.** Let $L = \overline{i_l, i_{l+r-1}}$, $L \subseteq C_{3.2} \subseteq \Omega$ and for each pair of jobs $(\alpha, \beta) \in L$ the conditions of Statement 10 are satisfied. Then the rearrangement of jobs of subset $L$ in the schedule on places $\overline{l, l+r-1}$ is opportune, in sense of (1), in the following mode: 1) beginning with place $l$, all jobs of subset $L \cap (A \backslash C)$ are placed in such a way that $\alpha \to \beta$, if $v_\alpha \geq v_\beta$, $\kappa_\alpha \geq \kappa_\beta$ and $\tau_{1\alpha} \leq \tau_{1\beta}$; 2) immediately after jobs of subset $L \cap (A \backslash C)$, the jobs of subset $L \cap C$ are placed in the schedule in an arbitrary mode; 3) immediately after jobs of subset $L \cap C$, the jobs of subset $L \cap (E \backslash C)$ are placed in the schedule in such a way that $\alpha \to \beta$, if $v_\alpha \geq v_\beta$, $\kappa_\alpha \geq \kappa_\beta$ and $\tau_{M\alpha} \geq \tau_{M\beta}$.

*Proof.* According to (10), relations $C_3 = C_{3.2} \subseteq (A \cup E)$ take place. At the same time, because of $L \subseteq C_{3.2}$, relations $L \subseteq (A \cup E)$ hold, too. The opportunity of ordering the categories of jobs of types $A$ or $E$ from $L$ in order $L \cap (A \backslash C) \to L \cap C \to L \cap (E \backslash C)$ results from Statement 9. Here, unlike conditions from Statement 11, the jobs of category $C$ are separated from subsets of jobs of types $A$ and $E$. With regard to the order of jobs of the same type $A \backslash C$ or $E \backslash C$, the conditions from Statement 11 coincide with those ones from Statement 10, if for pairs of jobs of type $A \backslash C$ ($(\alpha, \beta) \in A \backslash C$) to substitute the condition (14) by the $\tau_{1\alpha} \leq \tau_{1\beta}$ one and for pairs of jobs of type $E \backslash C$ ($(\alpha, \beta) \in E \backslash C$) to substitute the condition (14) by the $\tau_{M\alpha} \geq \tau_{M\beta}$ one; the relevancy of such substitutions is proved in paper [4]. According to Statement 8, the jobs of category $C$ can be placed in the schedule, on places between jobs of category $A \backslash C$ and those of category $E \backslash C$, in an arbitrary mode, that was required to be proved.

**Consequence 1**. If $L = C_{3.2} = \Omega$ and for each pair of jobs $(\alpha, \beta) \in L$ the conditions from Statement 10 are satisfied yet, then jobs ordering according to the modality, defined in Statement 11, results with an optimal, in sense of (1), schedule of all jobs from $\Omega$.

The relevancy of Consequence 1 results directly from Statement 11, taking into account that in this case $l = 1$ and $r = n$.

Evidently, the conditions from Statement 10 are not always satisfied for all jobs of type $C_{3.2}$. Two examples, for which the conditions from Statement 10 are satisfied for all jobs of type $C_{3.2}$, are described in Consequences 2 and 3.

**Consequence 2.** For the set of jobs, defined by relations (13), and in addition

$$\nu_i = \nu,\ \kappa_i = \kappa,\ i = \overline{1,n}, \tag{22}$$

the optimal, in sense of (1), ordering of all $n$ jobs is possible.

*Proof.* It is easy to observe that conditions (22), although correspond to the conditions (15) and (16) from Statement 10, are symmetric for each pair of jobs from the $n$ ones. So, the optimal, in sense of (1), ordering of the $n$ jobs depends on conditions (14) only. At the same time, conditions (14) can be substituted, according to [4], with: $\alpha \to \beta$ if $\tau_{1\alpha} \leq \tau_{1\beta}$ – for pairs of jobs of type $A$, and $\alpha \to \beta$ if $\tau_{M\alpha} \geq \tau_{M\beta}$ – for pairs of jobs of type $E$. In that way, there doesn't appear uncertainty with regard to the ordering of jobs of type $A$ and of type $E$ ones, that was required to be proved.

**Consequence 3.** If for $i \in A \subseteq C_{3.2}$ relations $\tau_{1i} \leq \tau_{1,i+1}$, $\nu_{1i} \geq \nu_{1,i+1}$, $k_{1i} \geq k_{1,i+1}$ take place and for $i \in E \subseteq C_{3.2}$ relations $\tau_{Mi} \geq \tau_{M,i+1}$, $\nu_{1i} \geq \nu_{1,i+1}$, $k_{1i} \geq k_{1,i+1}$ take place, then the optimal, in sense of (1), schedule of all the $n$ jobs is: $1 \to 2 \to 3 \to ... \to n-1 \to n$.

*Proof.* It is easy to observe that the conditions from Statement 10 are satisfied for each pair of the $n$ jobs defined in Consequence 3.

## 4 Ordering of $C_{3.1}$ type jobs

Consider a particular set $C_{3.1}$ of $n$ no decreasing (of type $A$) or no increasing (of type $E$) jobs with the following processing times:

$$\tau_{ji} = \begin{cases} \tau, j = \overline{1,\nu_i} \\ \theta_i, j = \overline{\nu_i + 1, \kappa_i} \\ \tau_{Mi}, j = \overline{\kappa_i + 1, M} \end{cases}, i = \overline{1,n}, \tag{23}$$

accepting, to extend the implicated categories of jobs, that there can be $\kappa_i = \nu_i$, too, when the job $i$ is only with two different processing times ($\theta_i$ and $\tau_{Mi}$). One example of two jobs $\alpha$ and $\beta$ of type $C_{3.1}$ is shown in Figure 2.

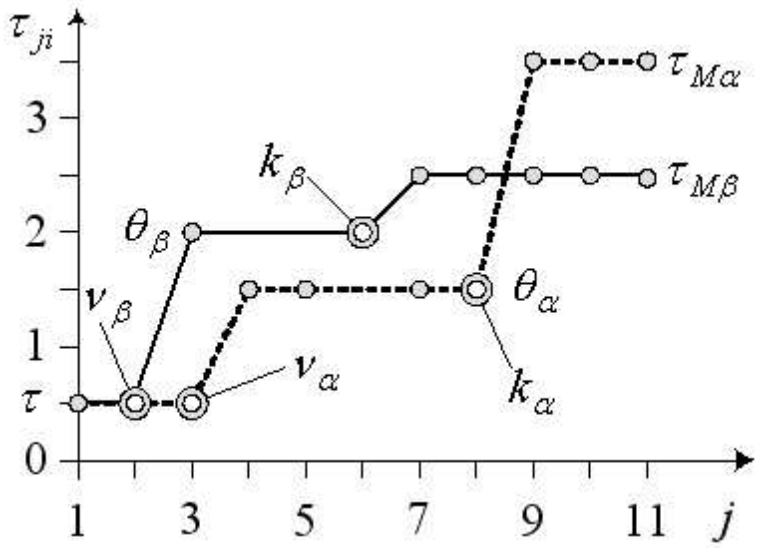


Figure 2. Two jobs $\alpha$ and $\beta$ of type $C_{3.1}$.

**Statement 12.** When placing jobs $(\alpha, \beta) \in C_{3.1}$ near each other in the schedule, it is opportune, in sense of (1), that $\alpha \to \beta$ if $\alpha \in A$ and $\beta \in E$ or if there take place the relations

$$\min(\theta_\alpha; \tau_{M\beta}) \leq \min(\tau_{M\alpha}; \theta_\beta) \tag{24}$$

and the conditions of one of the following cases:

$$a)\ \nu_\beta \leq \kappa_\beta \leq \nu_\alpha \leq \kappa_\alpha; \tag{25}$$

$$b)\ \nu_\beta \leq \nu_\alpha \leq \kappa_\beta \leq \kappa_\alpha \quad \text{and:} \quad (\alpha, \beta) \in A \text{ or } \{(\alpha, \beta) \in E; \theta_\alpha \geq \theta_\beta\}; \tag{26}$$

$$c)\ \nu_\beta \leq \nu_\alpha \leq \kappa_\alpha \leq \kappa_\beta \quad \text{and:} \quad \{(\alpha, \beta) \in A; \tau_{M\alpha} \leq \theta_\beta\} \text{ or } \{(\alpha, \beta) \in E; \tau_{M\alpha} \geq \theta_\beta\}. \tag{27}$$

*Proof.* According to (12), the jobs of $C_3 = C_{3.2}$ set are monotone no decreasing (belong to set $A$) or no increasing (belong to set $E$) ones. At the same time, according to Statement 7, if $\alpha \in A$ and $\beta \in E$ when placing jobs $\alpha$ and $\beta$ near each other in the schedule it is opportune, in sense of (1), that $\alpha \to \beta$. Thus it remains to prove the reliability of the statement for cases $(\alpha, \beta) \in A$ and $(\alpha, \beta) \in E$. The proof will be done by confirming the satisfaction of conditions (2)-(5) from Statement 1.

From (23), one can see that $\nu_\alpha \leq \kappa_\alpha$ and $\nu_\beta \leq \kappa_\beta$. That's why, for two concrete compared jobs $\alpha$ and $\beta$, there can be the same six variants (17) of relations among values $\nu_\alpha$, $\kappa_\alpha$, $\nu_\beta$ and $\kappa_\beta$ as at proving the Statement 10.

The procedure for confirming the satisfaction of conditions (2), for each of the six variants (17), is similar to that used when proving Statement 10, with the difference that, in place of relations (14), the relations (24) will be taken into account. The obtained results for the six variants (17), are described in Table 2. In this table, the cases for each of variants 1-6 are formed in a similar mode as the formation of analog cases when proving the Statement 10.

One can see from Table 2 that, unlike the jobs' set (13), for each of the six variants of the jobs' set, defined by relations (23), there exist many cases when the solutions by jobs type (local ones) differ; at the same time, solutions coincide or are larger for jobs of type $A$ ($(\alpha, \beta) \in A$), than for ones of type $E$ ($(\alpha, \beta) \in E$).

From the last column of Table 1, one can observe that the solutions of variants 1-5 are particular cases of the solution of **variant 6**. So, the general solution coincides with that of variant 6. As well, solutions by jobs type 1, 3, 7, 11, 15 and 19 (see penultimate column of Table 2) coincide with the solution of variants, to which these belong, hence are particular cases of the general solution; local solutions 21 and 22 coincide with the general one. While satisfying some supplementary conditions, there are other cases for which it is possible the partial ordering of jobs, too (see the penultimate column referring to numbered local solutions). It is easy to observe that a part of solutions by jobs types are particular cases of other local solutions or of the general one. The correspondence among these solutions is shown in Table 3.

Table 2. Cases that satisfy conditions (2) for the set $C_{3.2}$ of jobs

| Variant | Job type | Cases that satisfy local conditions (2) | Solution by job types | Solution for the variant |
|---|---|---|---|---|
| 1 | $(\alpha,\beta)\in A$ | 1.1-1.5 | 1. $\nu_\alpha=\kappa_\alpha=\nu_\beta=\kappa_\beta$ | $\nu_\alpha=\kappa_\alpha=\nu_\beta=\kappa_\beta$ |
| | | 1.1-1.5, 1.10 at $\tau_{M\alpha}\le\theta_\beta$ | 2. $\nu_\alpha=\kappa_\alpha=\nu_\beta\le\kappa_\beta$ | |
| | $(\alpha,\beta)\in E$ | 1.4, 1.7, 1.9, 1.10 | 3. $\nu_\alpha=\kappa_\alpha=\nu_\beta=\kappa_\beta$ | |
| | | 1.3, 1.4, 1.6-1.10 at $\tau_{M\alpha}\ge\theta_\beta$ | 4. $\nu_\alpha=\kappa_\alpha=\nu_\beta\le\kappa_\beta$ | |
| 2 | $(\alpha,\beta)\in A$ | 2.1-2.4, 2.8, 2.9 | 5. $\nu_\alpha=\nu_\beta\le\kappa_\alpha=\kappa_\beta$ | $\nu_\alpha=\nu_\beta=\kappa_\alpha=\kappa_\beta$ |
| | | 2.1-2.4, 2.8, 2.9, 2.10 at $\tau_{M\alpha}\le\theta_\beta$ | 6. $\nu_\alpha=\nu_\beta\le\kappa_\alpha\le\kappa_\beta$ | |
| | $(\alpha,\beta)\in E$ | 2.4, 2.7, 2.9, 2.10 | 7. $\nu_\alpha=\nu_\beta=\kappa_\alpha=\kappa_\beta$ | |
| | | 2.2, 2.4, 2.5, 2.7, 2.9, 2.10 la $\theta_\alpha\ge\theta_\beta$ | 8. $\nu_\alpha=\nu_\beta\le\kappa_\alpha=\kappa_\beta$ | |
| | | 2.2-2.10 at $\tau_{M\alpha}\ge\theta_\beta$ | 9. $\nu_\alpha=\nu_\beta\le\kappa_\alpha\le\kappa_\beta$ | |
| 3 | $(\alpha,\beta)\in A$ | 3.1-3.4, 3.8-3.10 | 10. $\nu_\alpha=\nu_\beta\le\kappa_\beta\le\kappa_\alpha$ | $\nu_\alpha=\nu_\beta=\kappa_\beta\le\kappa_\alpha$ |
| | $(\alpha,\beta)\in E$ | 3.3, 3,4, 3.6-3.10 | 11. $\nu_\alpha=\nu_\beta=\kappa_\beta\le\kappa_\alpha$ | |
| | | 3.2-3.10 at $\theta_\alpha\ge\theta_\beta$ | 12. $\nu_\alpha=\nu_\beta\le\kappa_\beta\le\kappa_\alpha$ | |
| 4 | $(\alpha,\beta)\in A$ | 4.1-4.9 | 13. $\nu_\beta\le\nu_\alpha\le\kappa_\alpha=\kappa_\beta$ | $\nu_\beta\le\nu_\alpha=\kappa_\alpha=\kappa_\beta$ |
| | | 4.1-4.10 at $\tau_{M\alpha}\le\theta_\beta$ | **14**. $\nu_\beta\le\nu_\alpha\le\kappa_\alpha\le\kappa_\beta$ | |
| | $(\alpha,\beta)\in E$ | 4.1, 4.4, 4.7, 4.9-4.10 | 15. $\nu_\beta\le\nu_\alpha=\kappa_\alpha=\kappa_\beta$ | |
| | | 4.1, 4.2, 4.4, 4.5, 4.7, 4.9, 4.10 at $\theta_\alpha\ge\theta_\beta$ | 16. $\nu_\beta\le\nu_\alpha\le\kappa_\alpha=\kappa_\beta$ | |
| | | 4.1-4.10 at $\tau_{M\alpha}\ge\theta_\beta$ | **17**. $\nu_\beta\le\nu_\alpha\le\kappa_\alpha\le\kappa_\beta$ | |
| 5 | $(\alpha,\beta)\in A$ | 5.1-5.10 | **18**. $\nu_\beta\le\nu_\alpha\le\kappa_\beta\le\kappa_\alpha$ | $\nu_\beta\le\nu_\alpha=\kappa_\beta\le\kappa_\alpha$ |
| | $(\alpha,\beta)\in E$ | 5.1, 5.3, 5.4, 5.6-5.10 | 19. $\nu_\beta\le\nu_\alpha=\kappa_\beta\le\kappa_\alpha$ | |
| | | 5.1-5.10 at $\theta_\alpha\ge\theta_\beta$ | **20**. $\nu_\beta\le\nu_\alpha\le\kappa_\beta\le\kappa_\alpha$ | |
| 6 | $(\alpha,\beta)\in A$ | 6.1-6.10 | **21**. $\nu_\beta\le\kappa_\beta\le\nu_\alpha\le\kappa_\alpha$ | $\nu_\beta\le\kappa_\beta\le\nu_\alpha\le\kappa_\alpha$ |
| | $(\alpha,\beta)\in E$ | 6.1-6.10 | **22**. $\nu_\beta\le\kappa_\beta\le\nu_\alpha\le\kappa_\alpha$ | |

From the last column of Table 3, one can observe that there are five different cases, defined by relations among values $\nu_\alpha$, $\kappa_\alpha$, $\nu_\beta$ and $\kappa_\beta$, which correspond to the general solution of variant 6 and to local solutions 14, 17, 18 and 20. These solutions correspond to cases (25)-(27), hence *conditions (2) are satisfied.*

It is needed still *to prove the satisfaction of conditions* (3)-(5). Case $\{\alpha\in E;\ \beta\in A\}$, which leads according to Statement 7 to the order $\beta\to\alpha$, secedes. Thus there remain cases that cover local solutions 14, 17, 18, 20, 21 and 22 from Table 2. According to Statement 4, these conditions take place for cases that are applicable at $(\alpha,\beta)\in A$ and, namely, those which cover the solutions 14, 18 and 21 from Table 2.

Table 3. Local solutions-particular cases of generalizing solutions for $C_{3.1}$ set

| Applicability domain | Local solutions-particular cases (from Table 2) | Generalizing solutions (from Table 2) |
|---|---|---|
| $(\alpha,\beta)\in A$ or $(\alpha,\beta)\in E$ | 1, 3, 7, 11, 15, 19, 21, 22 | Var.6. $\nu_\beta\le\kappa_\beta\le\nu_\alpha\le\kappa_\alpha$ |
| $(\alpha,\beta)\in A$ | 5, 10, 13 | 18. $\nu_\beta\le\nu_\alpha\le\kappa_\beta\le\kappa_\alpha$ |
| $(\alpha,\beta)\in E$ and $\theta_\alpha\ge\theta_\beta$ | 8, 12, 16 | 20. $\nu_\beta\le\nu_\alpha\le\kappa_\beta\le\kappa_\alpha$ |
| $(\alpha,\beta)\in A$ and $\tau_{M\alpha}\le\theta_\beta$ | 2, 6 | 14. $\nu_\beta\le\nu_\alpha\le\kappa_\alpha\le\kappa_\beta$ |
| $(\alpha,\beta)\in E$ and $\tau_{M\alpha}\ge\theta_\beta$ | 4, 9 | 17. $\nu_\beta\le\nu_\alpha\le\kappa_\alpha\le\kappa_\beta$ |

With regard to the other local cases (17, 20 and 22) from Table 2 applicable at $(\alpha,\beta)\in E$, at first it is needed to select server sequences, for which definition domains outlined by relations (3) and (4) are not empty. At $(\alpha,\beta)\in E$, on the base of relations (24), the inequality $\tau_{M\alpha}\ge\tau_{M\beta}$ takes place and, taking into account relation (23), the condition (3) can take place only in the frame of server sequence $j=\overline{\nu_\beta+1,\kappa_\beta}$. Let the condition (3) be satisfied, then the condition (4) can take place only in the frame of server sequence $j=\overline{1,\nu_\beta}$. Let the conditions (3) and (4) take place, then condition (5) is satisfied, too, because in the frame of server sequence $j=\overline{1,\nu_\beta}$, according to (23), equalities $\tau_{j\alpha}=\tau_{j\beta}=\tau$ take place, that was required to be proved.

**Statement 13.** For the set of $n$ jobs of type $C_{3.1}$, defined by relations (23), it is opportune, in sense of (1), that $\alpha\to\beta$ if $\alpha\in A$ and $\beta\in E$ or if there take place the relations (24) and conditions (25) at $(\alpha,\beta)\in E$ or conditions

$$\nu_\beta\le\nu_\alpha,\ \kappa_\beta\le\kappa_\alpha \text{ at } (\alpha,\beta)\in A. \tag{28}$$

*Proof.* It is easy to see that the conditions from Statement 13 are a subset of conditions from Statement 12. Therefore, conditions from Statement 13 satisfy conditions (2)-(5) from Statement 1. In that way, from the same considerations as when proving the Statement 10, it remains to prove that conditions (24), (25) at $(\alpha,\beta)\in E$ and (28) are

transitive.

The transitivity of conditions (24) is confirmed by Statement 3. With regard to conditions (25), let $\alpha \to \beta$ and $\beta \to \gamma$, then from the problem conditions we have: $\nu_\beta \leq \kappa_\beta \leq \nu_\alpha \leq \kappa_\alpha$ and $\nu_\gamma \leq \kappa_\gamma \leq \nu_\beta \leq \kappa_\beta$. Combining these two groups of inequalities, one can obtain $\nu_\gamma \leq \kappa_\gamma \leq \nu_\beta \leq \kappa_\beta \leq \nu_\alpha \leq \kappa_\alpha$, from where, eliminating factors $\nu_\beta$ and $\kappa_\beta$ referring to job $\beta$, results that inequalities $\nu_\gamma \leq \kappa_\gamma \leq \nu_\alpha \leq \kappa_\alpha$ take place, hence conditions (25) are transitive.

It remains to prove the transitivity of conditions (28). Let $(\alpha, \beta, \gamma) \in A$ and $\alpha \to \beta$, $\beta \to \gamma$, then from the problem conditions we have: $\nu_\beta \leq \nu_\alpha$, $\kappa_\beta \leq \kappa_\alpha$ and $\nu_\gamma \leq \nu_\beta$, $\kappa_\gamma \leq \kappa_\beta$. Combining in respective way these two pairs of inequalities, it is easy to obtain $\nu_\gamma \leq \nu_\beta \leq \nu_\alpha$ and $\kappa_\gamma \leq \kappa_\beta \leq \kappa_\alpha$, from where, eliminating factors $\nu_\beta$ and $\kappa_\beta$ referring to job $\beta$, results that inequalities $\nu_\gamma \leq \nu_\alpha$ and $\kappa_\gamma \leq \kappa_\alpha$ take place, hence conditions (28) are transitive, that was required to be proved.

**Consequence 4.** If relations $\tau_{1i} \leq \tau_{1,i+1}$, $\nu_{1i} \geq \nu_{1,i+1}$, $k_{1i} \geq k_{1,i+1}$ for $(i, i+1) \in A \subseteq C_{3.2}$ and relations $\tau_{Mi} \geq \tau_{M,i+1}$, $\theta_i \geq \theta_{i+1}$, $\nu_{1i} \geq \nu_{1,i+1}$, $k_{1i} \geq k_{1,i+1}$ for $(i, i+1) \in E \subseteq C_{3.2}$ take place, then the optimal, in sense of (1), ordering of all the $n$ jobs is: $i \in A \to j \in E$, $1 \to 2 \to 3 \to ... \to n-1 \to n$.

*Proof.* It is easy to observe that the conditions from Statement 12 are satisfied for any pair of the $n$ jobs defined by Consequence 3. The transitivity of conditions, defined in Consequence 4, can be confirmed in a similar mode as the conditions from Statement 13, that was required to be proved.

## 5 Ordering of $C_{3.3}$ type jobs

Let us consider a particular set $C_{3.3}$ of $n$ no decreasing (of type $A$) or no increasing (of type $E$) jobs with following processing times:

$$\tau_{ji} = \begin{cases} \tau_{1i}, j = \overline{1, \nu_i} \\ \theta_i, j = \overline{\nu_i + 1, \kappa_i} \\ \tau, j = \overline{\kappa_i + 1, M} \end{cases}, i = \overline{1, n}, \tag{29}$$

accepting, to extend implicated categories of jobs, that there can be $\kappa_i = \nu_i$, too, when job $i$ is only of two different processing times ($\tau_{1i}$ and $\theta_i$). One example of two jobs $\alpha$ and $\beta$ of type $C_{3.3}$ is shown in Figure 3.

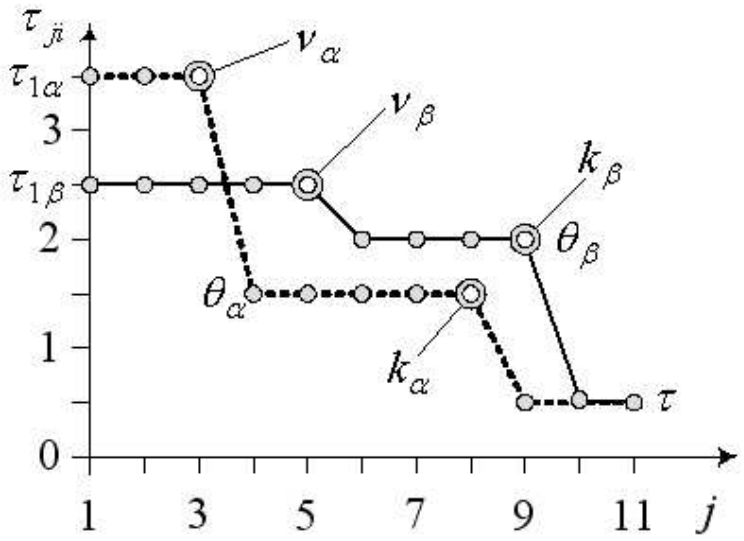


Figure 3. Two jobs $\alpha$ and $\beta$ of type $C_{3.3}$.

**Statement 14.** When placing jobs $(\alpha, \beta) \in C_{3.3}$ near each other in the schedule, it is opportune, in sense of (1), that $\alpha \to \beta$ if $\alpha \in A$ and $\beta \in E$ or if there take place the relations

$$\min(\tau_{1\alpha}; \theta_\beta) \le \min(\theta_\alpha; \tau_{1\beta}) \tag{30}$$

and the conditions of one of the following cases:

$$a)\ \nu_\beta \le \kappa_\beta \le \nu_\alpha \le \kappa_\alpha; \tag{31}$$

$$b)\ \nu_\beta \le \nu_\alpha \le \kappa_\beta \le \kappa_\alpha \quad \text{and:} \quad (\alpha, \beta) \in E \text{ or } \{(\alpha, \beta) \in A; \theta_\alpha \le \theta_\beta\}; \tag{32}$$

$$c)\ \nu_\alpha \le \nu_\beta \le \kappa_\beta \le \kappa_\alpha \quad \text{and:} \quad \{(\alpha, \beta) \in A; \theta_\alpha \le \tau_{1\beta}\} \text{ or } \{(\alpha, \beta) \in E; \theta_\alpha \ge \tau_{1\beta}\}. \tag{33}$$

*Proof.* According to (12), the jobs of set $C_3 = C_{3.3}$ are monotone no decreasing (of type $A$) or no increasing (of type $E$). At the same time, according to Statement 7, if $\alpha \in A$ and $\beta \in E$, then when placing jobs $\alpha$ and $\beta$ near each other in the schedule it is opportune, in sense of (1), that $\alpha \to \beta$. Thus it remains to prove the reliability of the statement for cases $(\alpha, \beta) \in A$ and $(\alpha, \beta) \in E$. Proof will be done by confirming the satisfaction of conditions (2)-(5) from Statement 1.

From (29), one can see that $\nu_\alpha \leq \kappa_\alpha$ and $\nu_\beta \leq \kappa_\beta$. Therefore, for two concrete compared jobs $\alpha$ and $\beta$, there can be the same six variants (17) of relations among values $\nu_\alpha$, $\kappa_\alpha$, $\nu_\beta$ and $\kappa_\beta$ as at Statement 10.

The procedure for the verification of satisfaction of the conditions (2) for each of the six variants (17) is similar to that used when proving Statement 10 with the difference that, in place of relations (14), the relations (30) are taken into account. The obtained results for the six variants (17) are described in Table 4. In this table, the cases for each of variants 1-6 are formed in a similar mode as the formation of analog cases when proving Statement 10.

From Table 4 it is easy to see that, unlike of jobs set (13), for each of the six variants of jobs set defined by relations (29), there exist many cases when the solution by job types differ; at the same time, this coincide or is larger for jobs of type $E$ ($(\alpha, \beta) \in E$), than for jobs of type $A$ ($(\alpha, \beta) \in A$).

Comparing the last column of Tables 2 and 4, it is easy to observe that the solutions of variants 1-6 for jobs set defined by relations (23) and the ones for jobs set defined by relations (29) coincide. At the same time, solutions by jobs type, specified in the penultimate column of Tables 2 and 4, don't always coincide.

From the last column of Table 4, one can see that the solutions of variants 1-5 are particular cases of the solution of **variant 6**. Thus, the general solution coincide with that of variant 6 one. As well, solutions by jobs types (local ones) 1, 3, 5, 10, 15 and 18 (see penultimate column of Table 4) coincide with solutions for variants, to which these belong, hence are particular cases of the general solution, and local solutions 21 and 22 coincide with the general one. At the same time, when satisfying some supplementary conditions, other cases, for which the

Table 4. Cases that satisfy conditions (2) for the jobs set (29)

| Variant | Job type | Cases that satisfy local conditions (2) | Solution by job types | Solution for the variant |
|---|---|---|---|---|
| 1 | $(\alpha,\beta)\in A$ | 1.1-1.5 | 1. $\nu_\alpha=\kappa_\alpha=\nu_\beta=\kappa_\beta$ | $\nu_\alpha=\kappa_\alpha=\nu_\beta=\kappa_\beta$ |
| | | 1.1-1.7 at $\theta_\alpha\le\tau_{1\beta}$ | 2. $\nu_\alpha\le\kappa_\alpha=\nu_\beta=\kappa_\beta$ | |
| | $(\alpha,\beta)\in E$ | 1.4, 1.7-1.10 | 3. $\nu_\alpha=\kappa_\alpha=\nu_\beta=\kappa\beta$ | |
| | | 1.1, 1.4, 1.7-1.10 at $\theta_\alpha\ge\tau_{1\beta}$ | 4. $\nu_\alpha\le\kappa_\alpha=\nu_\beta=\kappa_\beta$ | |
| 2 | $(\alpha,\beta)\in A$ | 2.1-2.4, 2.8 | 5. $\nu_\alpha=\nu_\beta=\kappa_\alpha=\kappa_\beta$ | $\nu_\alpha=\nu_\beta=\kappa_\alpha=\kappa_\beta$ |
| | | 2.1-2.4, 2.8, 2.9 at $\theta_\alpha\le\theta_\beta$ | 6. $\nu_\alpha=\nu_\beta\le\kappa_\alpha=\kappa_\beta$ | |
| | | 2.1-2.9 at $\theta_\alpha\le\tau_{1\beta}$ | 7. $\nu_\alpha\le\nu_\beta\le\kappa_\alpha=\kappa_\beta$ | |
| | $(\alpha,\beta)\in E$ | 2.2, 2.4, 2.5, 2.7, 2.9, 2.10 | 8. $\nu_\alpha=\nu_\beta\le\kappa_\alpha=\kappa_\beta$ | |
| | | 2.1, 2.2, 2.4, 2.5, 2.7, 2.9, 2.10 at $\theta_\alpha\ge\tau_{1\beta}$ | 9. $\nu_\alpha\le\nu_\beta\le\kappa_\alpha=\kappa_\beta$ | |
| 3 | $(\alpha,\beta)\in A$ | 3.1-3.4, 3.10 | 10. $\nu_\alpha=\nu_\beta=\kappa_\beta\le\kappa_\alpha$ | $\nu_\alpha=\nu_\beta=\kappa_\beta\le\kappa_\alpha$ |
| | | 3.1-3.4, 3.8-3.10 at $\theta_\alpha\le\theta_\beta$ | 11. $\nu_\alpha=\nu_\beta\le\kappa_\beta\le\kappa_\alpha$ | |
| | | 3.1-3.10 at $\theta_\alpha\le\tau_{1\beta}$ | **12.** $\nu_\alpha\le\nu_\beta\le\kappa_\beta\le\kappa_\alpha$ | |
| | $(\alpha,\beta)\in E$ | 3.2-3.10 | 13. $\nu_\alpha=\nu_\beta\le\kappa_\beta\le\kappa_\alpha$ | |
| | | 3.1-3.10 at $\theta_\alpha\ge\tau_{1\beta}$ | **14.** $\nu_\alpha\le\nu_\beta\le\kappa_\beta\le\kappa_\alpha$ | |
| 4 | $(\alpha,\beta)\in A$ | 4.1-4.8 | 15. $\nu_\beta\le\nu_\alpha=\kappa_\alpha=\kappa_\beta$ | $\nu_\beta\le\nu_\alpha=\kappa_\alpha=\kappa_\beta$ |
| | | 4.1-4.9 at $\theta_\alpha\le\theta_\beta$ | 16. $\nu_\beta\le\nu_\alpha\le\kappa_\alpha=\kappa_\beta$ | |
| | $(\alpha,\beta)\in E$ | 4.1, 4.2, 4.4, 4.5, 4.7, 4.9,4.10 | 17. $\nu_\beta\le\nu_\alpha\le\kappa_\alpha=\kappa_\beta$ | |
| 5 | $(\alpha,\beta)\in A$ | 5.1-5.7, 5.10 | 18. $\nu_\beta\le\nu_\alpha=\kappa_\beta\le\kappa_\alpha$ | $\nu_\beta\le\nu_\alpha=\kappa_\beta\le\kappa_\alpha$ |
| | | 5.1-5.10 at $\theta_\alpha\le\theta_\beta$ | **19.** $\nu_\beta\le\nu_\alpha\le\kappa_\beta\le\kappa_\alpha$ | |
| | $(\alpha,\beta)\in E$ | 5.1-5.10 | **20.** $\nu_\beta\le\nu_\alpha\le\kappa_\beta\le\kappa_\alpha$ | |
| 6 | $(\alpha,\beta)\in A$ | 6.1-6.10 | **21.** $\nu_\beta\le\kappa_\beta\le\nu_\alpha\le\kappa_\alpha$ | $\nu_\beta\le\kappa_\beta\le\nu_\alpha\le\kappa_\alpha$ |
| | $(\alpha,\beta)\in E$ | 6.1-6.10 | **22.** $\nu_\beta\le\kappa_\beta\le\nu_\alpha\le\kappa_\alpha$ | |

partial ordering of jobs is possible, exist, too (see penultimate column referring to numbered solutions by jobs types – local ones). One can easily observe that a part of solutions by jobs types are particular cases of other local solutions or of the general solution. The correspondence among them is shown in Table 5.

Table 5. Local solutions-particular cases of generalizing solutions for $C_{3.3}$ set

| Applicability domain | Local solutions-particular cases (from Table 4) | Generalizing solutions (from Table 4) |
|---|---|---|
| $(\alpha,\beta)\in A$ or $(\alpha,\beta)\in E$ | 1, 3, 5, 10, 15, 18, 21, 22 | Var.6. $\nu_\beta\le\kappa_\beta\le\nu_\alpha\le\kappa_\alpha$ |
| $(\alpha,\beta)\in E$ | 8, 13, 17 | 20. $\nu_\beta\le\nu_\alpha\le\kappa_\beta\le\kappa_\alpha$ |
| $(\alpha,\beta)\in A$ and $\theta_\alpha\le\theta_\beta$ | 6, 11, 16 | 19. $\nu_\beta\le\nu_\alpha\le\kappa_\beta\le\kappa_\alpha$ |
| $(\alpha,\beta)\in E$ and $\theta_\alpha\ge\tau_{1\beta}$ | 4, 9 | 14. $\nu_\alpha\le\nu_\beta\le\kappa_\beta\le\kappa_\alpha$ |
| $(\alpha,\beta)\in A$ and $\theta_\alpha\le\tau_{1\beta}$ | 2, 7 | 12. $\nu_\alpha\le\nu_\beta\le\kappa_\beta\le\kappa_\alpha$ |

Thus, there are five different cases, defined by relations among values $\nu_\alpha$, $\kappa_\alpha$, $\nu_\beta$ and $\kappa_\beta$, that correspond to the general solution of variant 6 and to local solutions 12, 14, 19 and 20. These correspond to cases (31)-(33), hence *conditions (2) are satisfied.*

We have still *to prove the satisfaction of conditions* (3)-(5). The case $\{\alpha\in E;\beta\in A\}$, which leads, according to Statement 4, to the order $\beta\to\alpha$, secedes. Thus there remain cases that cover solutions 12, 14, 19, 20, 21 and 22 from Table 4. According to Statement 4, conditions (3)-(5) take place for cases applicable at $(\alpha,\beta)\in A$, namely that which cover solutions 12, 19 and 21 from Table 4.

With regard to the other cases from Table 4 (14, 20 and 22) applicable at $(\alpha,\beta)\in E$, firstly it is needed to select server sequences, for which definition domains, outlined by relations (3) and (4), are not empty. At $(\alpha,\beta)\in E$, on the basis of relations (30), the inequality $\theta_\alpha\ge\theta_\beta$ takes place and, on the basis of conditions (31)-(33), the inequality $\kappa_\beta\le\kappa_\alpha$ takes place. Thus, taking into account relation (23), the condition (3) can take place only in the frame of servers sequence $j=\overline{1,\nu_\beta}$: (a) at $\nu_\beta>\nu_\alpha$ and $\theta_\alpha<\tau_{1\beta}$ or (b) at $\nu_\beta\le\nu_\alpha$ and, respectively, $\tau_{1\alpha}<\tau_{1\beta}$. In the first of these two cases, according to data

from Table 4, the condition $\theta_\alpha < \tau_{1\beta}$ doesn't hold for local solution 14, but can take place for local solutions 20 and 22. At the same time, according to data from Table 4, for local solutions 20 and 22 the inequality $\nu_\beta \leq \nu_\alpha$ takes place, hence case (a) can't take place. Let the case (b) take place and the condition (3) is satisfied; then the condition (4) can't be held, because the condition (3) takes place for the entire servers sequence $j = \overline{1,\nu_\beta}$. Thus, conditions (3) and (4) don't take place concomitantly; hence conditions (3)-(5) are satisfied that was required to be proved.

**Statement 15.** For the set of $n$ jobs of type $C_{3.3}$, defined by relations (29), it is opportune, in sense of (1), that $\alpha \to \beta$ if $\alpha \in A$ and $\beta \in E$ or if there take place the relations (30) and conditions (31) at $(\alpha, \beta) \in A$ or conditions

$$\nu_\beta \leq \nu_\alpha,\ \kappa_\beta \leq \kappa_\alpha \text{ at } (\alpha, \beta) \in E. \tag{34}$$

*Proof.* We can see that the conditions from Statement 15 are a subset of the ones from Statement 14. Therefore, the conditions from Statement 15 satisfy the conditions (2)-(5) from Statement 1. Thus, from the same considerations as when proving the Statement 10, it remains to prove, that conditions (30), (31) at $(\alpha, \beta) \in A$ and (34) are the transitive ones.

The transitivity of relations (30) is confirmed by Statement 3. With regard to conditions (31), these coincide with the (25) ones and the transitivity of the last are proved in Statement 13. Note, that the proof of transitivity of conditions (34) doesn't depend on the class ($A$ or $E$) to which the jobs $\alpha$ and $\beta$ belong. At the same time, if not to take into account the class to which the jobs $\alpha$ and $\beta$ belong, then relations (34) coincide with those from (28) and the transitivity of last ones is confirmed by Statement 13, that was required to be proved.

**Consequence 5.** If the relations $\tau_{1i} \leq \tau_{1,i+1}$, $\theta_i \leq \theta_{i+1}$, $\nu_{1i} \geq \nu_{1,i+1}$, $k_{1i} \geq k_{1,i+1}$ for $(i, i+1) \in A \subseteq C_{3.3}$ and relations $\tau_{Mi} \geq \tau_{M,i+1}$, $\nu_{1i} \geq \nu_{1,i+1}$, $k_{1i} \geq k_{1,i+1}$ for $(i, i+1) \in E \subseteq C_{3.3}$ take place, then the optimal, in sense of (1), ordering of all the $n$ jobs is: $i \in A \to j \in E$, $1 \to 2 \to 3 \to ... \to n-1 \to n$.

*Proof.* We can easily observe that the conditions from Statement 14 are satisfied for each pair from the $n$ jobs, defined in Consequence 5. At the same time, the transitivity of relations, defined in Consequence 5, can be confirmed in the same mode as of ones from the Statement 15, that was required to be proved.

## 6 Conclusions

Three classes $C_{3.1}$, $C_{3.2}$ and $C_{3.3}$ of systems with monotone jobs of no more than three different processing times in the *Mxn* Bellman-Johnson ordering problem are investigated. For the class $C_{3.2}$, it is obtained a set of relatively simple rules for partial ordering or, in the case that all jobs satisfy the respective conditions, total ordering of the $n$ jobs. There are obtained the rules for ordering in pairs of adjacent jobs for classes of systems $C_{3.1}$ and $C_{3.3}$, too. Rules for ordering the pairs of jobs, when placing them anywhere in the schedule are defined, too. Examples of concrete systems, for which the optimal order of all $n$ jobs can be obtained, are done, too. The obtained results can be used for jobs ordering in sequential systems, aiming to minimize the total processing time of all jobs.

## References

[1] S.M.Johnson. *Optimal Two- and Three-Stage Production Schedules with Setup Times Included*// Naw. Res. Log. Quart. Vol. 1. nr. 1, 1954. – pp. 61–68.

[2] R.W. Conway, W.L.Maxwell, L.W.Miller. *Theory of Scheduling.* – New York: Addison-Wesley, 1967.

[3] V.S.Tanaev, V.V.Shcurba. *Introduction to theory of scheduling.* – Moscow: Nauka, 1975 (Russian).

[4] I.Bolun. *A modification of the Johnson's ordering algorithm*// Proceedings of the republican seminar ,,Systems and means for the

integrated processing of information". – Chisinau: ICSP, 1981. – pp. 43-46 (Russian).

[5] I.Bolun. *On a problem of the ordering of jobs in multiphase systems*// Models and algorithms for informatics systems. Chisinau: Stiinta, 1986. – pp. 29–58 (Russian).

[6] I.M.Artamonov. *Referring an algorithm for solving the Bellman-Johnson problem*// Models and algorithms for solving planning and control problems de. - Chiinu: Stiinta, 1982 (Russian).

[7] I.Bolun. *Ordering of monotone jobs in the Mxn Bellman-Johnson problem*// Economica, nr. 1(57)/2007. Chisinau: Editura ASEM (Romanian).

[8] I.Bolun. *Ordering of monotone jobs with no more than two different processing times in sequential systems*// Annals of the Academy of Economic Studies from Moldova, Vol. 5. - Chisinau: Editura ASEM, 2007 (Romanian).

Ion Bolun,

Received April 18, 2007

Academy of Economic Studies from Moldova,
Cybernetics, Statistics and Economic Informatics School
E–mail: *bolun@ase.md*

# Abstracts of Doctor Habilitatus Thesis

**Title:** Intelligent interfaces for computer algebra systems
**Author:** Svetlana Cojocaru
**Supervisor:** Prof. Constantin Gaindric
**Institute:** Institute of Mathematics and Computer Science, Academy of Sciences of Moldova, Chişinău
**Date of defence:** February, 9, 2007

The dissertation is devoted to the investigation of human–computer interaction modalities in computer algebra systems and developing of methods for intelligent interfaces creation for such systems.

The principles of interfaces design are investigated, intelligent interfaces and interfaces for computer algebra systems are classified; the intelligent interfaces features are described. Some aspects of natural language usage in intelligent interfaces are revealed.

The problem of computational lexicon development for inflectional languages is investigated. A notion of inflectional grammar is proposed; it permits to describe the inflexion process in the case when the inflexion model is known. This grammar was applied to formalize the inflexion process in Romanian. An algorithm of the morphological models ascertainment for the cases when the corresponding models are not known is proposed.

According to the proposed methods (static and dynamic ones) a computational lexicon containing about 1 million words was elaborated. It was used for developing of several applications.

A number of methods to implement intelligence features in interfaces for computer algebra systems are proposed. They include problems interception from the user, adaptation to his preferences, error prevention.