PRACE CO PAN • CC PAS REPORTS

Józef Winkowski

A language for describing non-sequential processes

253

1976

WARSZAWA

CENTRUM OBLICZENIOWE POLSKIEJ AKADEMII NAUK
COMPUTATION CENTRE POLISH ACADEMY OF SCIENCES
00-901 WARSAW, P.O. Box 22, POLAND

Józef Winkowski

A LANGUAGE FOR DESCRIBING NON-SEQUENTIAL PROCESSES

253

Warsaw 1976

Mailing address: dr Józef Winkowski
Computation Centre PAS
P.O. Box 22
00 -901 Warsaw
POLAND

P r i n t e d   a s   a   m a n u s c r i p t
N a   p r a w a c h   r ę k o p i s u

Nakład 700 egz. Ark. wyd. 0,85; ark. druk. 1,375. Papier offset. kl. III, 70 g, 70 x 100. Oddano do druku w maju 1976 r. W. D. N. Zam. nr 400/0/76
J-113

AMS CATEGORIES: 68A05
CS CATEGORIES: 5.24

Abstract . Содержание . Streszczenie

In the paper a language is suggested for describing non-sequential processes. Mathematical semantics of two types are formulated for this language and relationships between such semantics are explained.

Язык описания параллельных процессов

В статье предлагается язык описания параллельных процессов. Для этого языка определяются математические семантики двух типов и выявляются соотношения между такими семантиками.

Język opisu procesów niesekwencyjnych

W pracy zaproponowano język opisu procesów niesekwencyjnych. Dla tego języka zdefiniowano semantyki dwóch typów i zbadano związki, jakie zachodzą między takimi semantykami.

1. INTRODUCTION

The approach we present here is inspired by some ideas of Petri[9,10], Genrich[3], Mazurkiewicz[7], and others. It comes from looking at any process as at changing relations corresponding to some predicate symbols.

We concentrate on the processes which run according to some rules from some finite sets called algorithms. Each rule r consists of two finite sets r_1, r_2 of atomic formulas of the form

$$\omega(x_1,\ldots,x_{a(\omega)})$$

where ω is a predicate symbol of the arity $a(\omega)$, and $x_1,\ldots,$ $x_{a(\omega)}$ are some variables. It applies (may be concurrently with other rules) in such cases in which there is a one-to-one correspondence between the variables of the formulas of r and some objects which actually exist or may appear such that:

(1) all the formulas of r_1 are satisfied by the corresponding objects which exist,

(2) no formula of $r_2 \setminus r_1$ is satisfied,

(3) only appearing objects correspond to the variables which occur in the formulas of r_2 but do not occur in the formulas of r_1.

The application leads to a change after which the conditions corresponding to the formulas of $r_1 \setminus r_2$ cease, and the conditions corresponding to the formulas of $r_2 \setminus r_1$ start to be satisfied. Other satisfied conditions remain the same. The process continues while some rules apply. Otherwise it terminates.

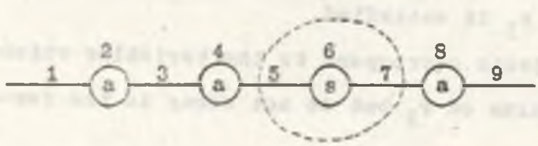
Example 1 The production $s \to sa$ of a context-free grammar is a rule. It may be illustrated as

and written as $r=(r_1,r_2)$ with

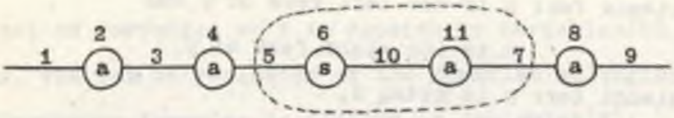
r_1={y is at the right end of x, y is at the left end of z,
y is an occurrence of s}.

r_2={y is at the right end of x, y is at the left end of u,
v is at the right end of u, v is at the left end of z,
y is an occurrence of s, v is an occurrence of a}

This rule applies in the cases of derivation processes where we have derived a word with an occurrence of s. For instance, in the case



the rule applies and the application leads to the word



with a new link 10 and a new occurrence 11 of a.

Example 2 The labelled instruction

$$e: Y := F(X)$$

is the rule $r=(r_1,r_2)$ with

$r_1=\{$the value of e is x, y follows x, the control is at x, the value of X is u, the value of Y is v, $F(u)=w\}$

$r_2=\{$the value of e is x, y follows x, the control is at y, the value of X is u, the value of Y is w, $F(u)=w\}$

A program can be considered as a finite set of such rules, i.e., as an algorithm.

Example 3 (after Dijkstra [1]) There are five philosophers sitting at a round table. They are alternately thinking or eating something with two forks which they share with their neighbours as it is shown in the picture.

If $R(x,y,z)$ stands for: x is the left fork of y and
z is the right fork of y,
$H(y,x)$ stands for: y is using x,
$F(x)$ stands for: x is free,
then $R(10,1,2)$, $R(2,3,4)$, $R(4,5,6)$, $R(6,7,8)$, $R(8,9,10)$, and the philosophers behave according to the following rules:

$$(\{R(x,y,z),F(x)\},\ \{R(x,y,z),H(y,x)\})$$
$$(\{R(x,y,z),F(z)\},\ \{R(x,y,z),H(y,z)\})$$
$$(\{R(x,y,z),H(y,x),H(y,z)\},\ \{R(x,y,z),F(x),F(z)\})$$

The idea to describe processes by finite sets of rules of the above type seems to be quite universal. It is the key idea of our approach.

2. THE LANGUAGE

Now we define our language for describing non-sequential processes. This language (of algorithms) is determined (up to inessential syntactic details) by defining algorithms in a formal way. The basic definitions are the following.

An <u>elementary formula</u> is an ordered $(a(\omega)+1)$-tuple

$$f = (\omega, x_1, \ldots, x_{a(\omega)})$$

where ω is a predicate symbol of the arity $a(\omega)$, and $x_1, \ldots, x_{a(\omega)}$ are some variables. Such a formula is written as

$$\omega(x_1, \ldots, x_{a(\omega)})$$

The set of variables of f is denoted by Variables(f). More generally, the set of variables of the formulas belonging to a set F of elementary formulas is denoted by Variables(F).

A <u>rule</u> is an ordered pair

$$r = (r_1, r_2)$$

of two different finite sets r_1, r_2 of elementary formulas. Such a rule is usually written as

$$r_1 \longrightarrow r_2$$

The set r_1 is said to be the <u>left part</u> of r and is denoted by L(r). The set r_2 is said to be the <u>right part</u> of r and is denoted by R(r).

An <u>algorithm</u> is a finite set of rules.

3. SEMANTICS

A semantics of the language of algorithms can be given by assigning a class of processes to every algorithm. The processes of this class correspond to possible executions of the algorithm. To characterize them we introduce some preliminary notions.

An <u>elementary situation</u> is an ordered $(a(\omega)+1)$-tuple

$$s = (\omega, b_1, \ldots, b_{a(\omega)})$$

where ω is a predicate symbol of the arity $a(\omega)$, and $b_1, \ldots, b_{a(\omega)}$ are some objects. Such a situation is written as

$$\omega(b_1, \ldots, b_{a(\omega)})$$

and it means that the objects $b_1,\dots,b_{a(\omega)}$ are in the relation corresponding to the predicate symbol ω. The set of these objects is denoted by Objects(s).

A <u>situation</u> is a set S of elementary situations. The set of objects which occur in the elementary situations belonging to S is denoted by Objects(S).

An <u>elementary change</u> is an ordered pair

$$m = (m_1, m_2)$$

of two finite sets m_1, m_2 of elementary situations. Such a change is written as

$$m_1 \longrightarrow m_2$$

The set m_1 is said to be the <u>left part</u> of m and is denoted by L(m). The set m_2 is said to be the <u>right part</u> of m and is denoted by R(m). The elementary change m is said to be <u>possible</u> in a situation S if there is in S no elementary situation from $R(m) \setminus L(m)$ and if no object of $\mathrm{Objects}(R(m)) \setminus \mathrm{Objects}(L(m))$ belongs to Objects(S). A situation S' is said to be the <u>result</u> of the change m in the situation S iff m is possible in S and $S' = (S \setminus L(m)) \cup R(m)$.

Elementary changes m,n are said to be in a <u>conflict</u> if $L(m) \setminus R(m) \neq L(n) \setminus R(n)$ or $R(m) \setminus L(m) \neq R(n) \setminus L(n)$.

An <u>instance of an elementary formula</u>

$$f = \omega(x_1,\dots,x_{a(\omega)})$$

is an elementary situation

$$s = \omega(b_1,\dots,b_{a(\omega)})$$

such that there is a one-to-one mapping φ of Variables(f) onto Objects(s) with $\varphi(x_i)= b_i$ for $i= 1,\dots,a(\omega)$. The mapping φ is said to be a <u>realisation</u> of f in s.

An <u>instance of a set F of elementary formulas</u> is a situation S such that there is a one-to-one mapping φ of Variables(F) onto Objects(S), and $\omega(b_1,\dots,b_{a(\omega)})\in S$ iff $b_1= \varphi(x_1),\dots,$ $b_{a(\omega)}=\varphi(x_{a(\omega)})$ for some $\omega(x_1,\dots,x_{a(\omega)})\in F$. The mapping φ is said to be a <u>realisation</u> of F in S. Of course, every situation $\omega(\varphi(x_1),\dots,\varphi(x_{a(\omega)}))\in S$ is then an instance of the formula $\omega(x_1,\dots,x_{a(\omega)})\in F$. We call it the <u>instance of</u> $\omega(x_1,\dots,x_{a(\omega)})$ <u>in the instance S of F</u>.

An <u>instance of a rule</u> r is an elementary change m such that:

(1) there is a one-to-one mapping φ of Variables($L(r)\cup R(r)$) onto Objects($L(m)\cup R(m)$),

(2) $\omega(b_1,\dots,b_{a(\omega)})\in L(m)$ iff $b_1=\varphi(x_1),\dots,b_{a(\omega)}=\varphi(x_{a(\omega)})$ for some $\omega(x_1,\dots,x_{a(\omega)})\in L(r)$,

(3) $\omega(b_1,\dots,b_{a(\omega)})\in R(m)$ iff $b_1=\varphi(x_1),\dots,b_{a(\omega)}=\varphi(x_{a(\omega)})$ for some $\omega(x_1,\dots,x_{a(\omega)})\in R(r)$.

The mapping φ is said to be a <u>realisation</u> of r in m. The rule r is said to be <u>applicable</u> in a situation S iff there is an instance m of r which is possible in S.

Now we are ready to define executions of algorithms.

By an <u>execution</u> of an algorithm A we mean any ordered quintuple

$$E= (T,U,\mathrm{pre},\mathrm{post},F)$$

such that:

(E1) T is a non-empty set (of <u>occurrences</u> of elementary situations), (E2) U is a set (of <u>occurrences</u> of elementary changes),

(E3) $pre \subseteq T \times U$ is a binary relation (if $(t,u) \in pre$ then the occurrence t of an elementary situation is said to be a <u>precondition</u> of the occurrence u of an elementary change),

(E4) $post \subseteq U \times T$ is a binary relation (if $(u,t) \in post$ then the occurrence t of an elementary situation is said to be a <u>postcondition</u> of the occurrence u of an elementary change),

(E5) F is a function that assigns the elementary situation F(t) to every occurrence t of this situation, and the elementary change F(u) to every occurrence u of this change,

(E6) $s \in L(F(u))$ iff $s = F(t)$ for some t with $(t,u) \in pre$,
$s \in R(F(u))$ iff $s = F(t)$ for some t with $(u,t) \in post$,
and F is locally one-to-one, i.e., $F(t) \neq F(t')$ for every u and t,t' with $t \neq t'$, $(t,u) \in pre$ or $(u,t) \in post$, and $(t',u) \in pre$ or $(u,t') \in post$,

(E7) the reflexive and transitive closure of the following relation R in T:
tRt' iff $t = t'$ or $(t,u) \in pre$, $(u,t) \notin post$, $(t',u) \notin pre$, $(u,t') \in post$ for some u
is an ordering $\leqslant$ of T,

(E8) $t \not\leqslant t'$, $t' \not\leqslant t$, $t \neq t'$ implies $F(t) \neq F(t')$ for $t \in T$, $t' \in T$,

(E9) every non-empty subset of T has a minimal element,

(E10) every non-empty subset of T with an upper bound has a maximal element,

(E11) for every $u \in U$ the elementary change $F(u)$ is a possible instance of a rule of A; this can be precisely formulated as follows:
we say that two elementary situation occurrences $t \in T$, $t' \in T$ are <u>independent</u> (or potentially <u>concurrent</u>) iff neither $t \leqslant t'$ nor $t' \leqslant t$; every maximal set of independent occurrences of elementary situations is said to be a <u>case</u>; to every case c the set $F(c)$ of the elementary situations $F(t)$ with $t \in c$ corresponds and this set is a situation; the condition is:
for every elementary change occurrence u with the preconditions in a case c there is a rule r of A such that $F(u)$ is an instance of r and this instance is a change which is possible in the situation $F(c)$,

(E12) if two different elementary change occurrences u,v have a common precondition (postcondition) t then t must be a postcondition (precondition) of u and v (this means that it is decided in any case which of possible conflict changes occur),

(E13) the execution terminates iff no rule of A can be applied; this can be formulated as follows:
let c_{max} be the set of maximal elements of T (it is a case); if an instance m of a rule of A is possible in the situation $F(c_{max})$ then m has an occurrence with all the preconditions in c_{max} and no postcondition, or is in a conflict with an elementary change having an occurrence whose preconditions are in c_{max}.

The conditions (E1)-(E13) characterize the class of all possible executions of A. When added (together with a characterization of A) to the usual set theory they constitute an extension of the set theory. This extension is said to be a <u>general objective semantics</u> of A and is denoted by SEM(A). The assignment SEM: $A \longmapsto SEM(A)$ is said to be a <u>general objective semantics of the language of algorithms</u>. The semantics are said to be objective because they characterize the considered processes in terms of objects which really exist in these processes.

When we employ in our algorithms some arithmetical or other notions the general objective semantics can be extended to appropriate <u>special</u> ones. This can be done by adding to every of the theories SEM(A) some specific axioms which specify the meaning we have in mind to some predicate symbols. For instance, we can specify $R(\vec{a},\vec{b},c)$ as $c= a+b$ by adding the axiom

$$(\forall t)(F(t)= R(a,b,c) \Longrightarrow c= a+b)$$

Of course, this may lead to some inconsistent semantics.

4. SUBJECTIVE SEMANTICS

It is sometimes convenient to consider executions of an algorithm A from the point of view of an observer. This leads to a new semantics which is said to be <u>subjective</u>.

If elementary changes are instantaneous the observer observes a sequence $\{G(p)\}_{p\in P}$ of <u>global states</u> with P being the set Nat of natural numbers (if the execution does not terminate) or an initial segment of Nat (if the execution terminates). The global states are some more or less complex situations. The transition from $G(p)$ to $G(p+1)$ is considered as consisting of concurrent applications of some rules of A in the situation $G(p)$. From the point of view of the observer the indices $p \in P$ are <u>phases</u> of the observed execution. Which global states correspond to the consecutive phases depends on some unobservable factors. In this way parallelism is replaced by indeterminism.

If elementary changes are time-consuming it may be that global states are not directly observable objects. However, the observer can note the elementary situations which arise when elementary changes terminate, and erease the elementary situations which cease. Then all the elementary situations which may be considered as actual ones constitute something that corresponds to a global state. Thus, the observer observes again a sequence $\{G(p)\}_{p \in P}$ of global states in his registration. Transitions between consecutive global states are now results of terminations of elementary changes but still they may be considered as consisting of concurrent applications of some rules of the algorithm. In consequence, we have the same description as before with a slightly modified interpretation.

What we have said enables us to define executions of algorithms from the point of view of an observer. Namely, by an <u>execution</u> of an algorithm A we mean now any ordered quadruple

$$E' = (P, G, \mathrm{Possible}, \mathrm{Occurs})$$

such that:

(E′1) P is the set Nat of natural numbers or an initial segment of Nat; elements of P are said to be phases of E′,

(E′2) G is a mapping that assigns a global state G(p) of E′ to every phase p,

(E′3) Possible is the following binary relation: Possible(m,p) iff m is an instance of a rule of A, $p \in P$, and m is possible in G(p),

(E′4) Occurs is a binary relation contained in Possible (Occurs(m,p) means that the elementary change m occurs exactly at the phase p),

(E′5) there is no instance m of a rule of A, and $q \in P$, such that Possible(m,p) for all $p \geqslant q$ (the execution does not terminate if some rules of A are applicable),

(E′6) if Occurs(m,p) for some m then $p+1 \in P$,

(E′7) if $p \in P$ and $p+1 \in P$ then there is a non-empty set M of instances of rules of A such that Occurs(m,p) for every $m \in M$, and $G(p+1) = (G(p) \setminus \bigcup_{m \in M} L(m)) \cup \bigcup_{m \in M} R(m)$ (this is a characterization of the transition from the phase p to the next phase p+1),

(E′8) if Occurs(m,p) and Occurs(m′,p) with $m \neq m'$ then $L(m) \cap L(m') \subsetneq R(m) \cap R(m')$ and $R(m) \cap R(m') \subsetneq L(m) \cap L(m')$ (i.e., conflicts are decided).

Adding the axioms (E'1)-(E'8) and a characterization of A to the set theory gives a theory which is said to be a general subjective semantics of A and is denoted by sem(A). The correspondence sem: $A \longmapsto sem(A)$ is said to be a general subjective semantics of the language of algorithms. The general subjective semantics can be extended to special ones by adding appropriate axioms, just as in the case of objective semantics.

Subjective semantics were employed more or less explicitly in a number of papers concerning non-sequential processes (Karp, Miller[4], Milner[8], Mazurkiewicz[6]). They are very handy tools to investigate processes because they allow one to apply the powerful methods which have been developed for sequential processes. In particular, the well known method of invariants (Mazurkiewicz[5]) can be exploited. The problem only arises whether subjective semantics are powerful enough for describing and proving properties of non-sequential processes. This problem has been answered positively for a class of processes in Winkowski[12]. In what follows we give a solution of it for the considered executions of algorithms. This solution bases on the method of modelling that was described in Winkowski[11,12].

5. MODELLING OBJECTIVE SEMANTICS IN SUBJECTIVE ONES

By modelling of a theory T in another theory T' we mean an assignment μ of formulas (and terms) of T' to the formulas (terms) of T that preserves free variables (term variables), logical operations (substitutions), and theorems. If such a modelling exists then all the theorems of T can be interpreted and proved in T'. Thus, every model of T' has all the properties of models of T which can be formulated in T. In particular, having such a model we can construct a model of T. Hence, T is consistent if only T' is consistent.

To construct a modelling of the objective semantics SEM(A) of an algorithm A in the subjective semantics sem(A) we take the identity modelling of the set theory with a characterization of A in itself and extend it to a correspondence μ_A between formulas (and constants) of SEM(A) and those of sem(A). Namely, we define:

$\mu_A(t \in T)$ as

$$(\exists x)(\exists p)(\exists q)(t = (x,p,q) \,\&\, x \text{ is an elementary situation } \&$$
$$p \in P \,\&\, q \in P \,\&\, p \leq q \,\&\, (\forall k: p \leq k \leq q)(x \in G(k)) \,\&$$
$$(p-1 \notin P \vee p-1 \in P \,\&\, x \notin G(p-1)) \,\&$$
$$(q+1 \in P \,\&\, x \notin G(q+1))) \vee$$
$$(\exists x)(\exists p)(t = (x,p) \,\&\, x \text{ is an elementary situation } \& \, p \in P \,\&$$
$$(\forall k \in P: p \leq k)(x \in G(k)) \,\&$$
$$(p-1 \notin P \vee p-1 \in P \,\&\, x \notin G(p-1)))$$

$\mu_A(u \in U)$ as $(\exists m)(\exists p)(u = (m,p) \,\&\, \mathrm{Occurs}(m,p))$

$\mu_A((t,u) \in pre)$ as

$$\mu_A(t \in T) \,\&\, \mu_A(u \in U) \,\&$$
$$(\exists p)(\exists q)(\exists x)(\exists m)(t = (x,p,q) \,\&\, u = (m,q) \,\&\, x \in L(m))$$

$\mu_A((u,t) \in post)$ as

$$\mu_A(t \in T) \,\&\, \mu_A(u \in U) \,\&$$
$$(\exists p)(\exists m)(u = (m,p) \,\&$$
$$((\exists x)(t = (x,p+1) \,\&\, x \in R(m)) \vee$$
$$(\exists x)(\exists q)(t = (x,p+1,q) \,\&\, x \in R(m)))$$

$\mu_A(y = F(x))$ as

$$\mu_A(x \in T) \,\&\, ((\exists p)(x = (y,p)) \vee (\exists p)(\exists q)(x = (y,p,q))) \vee$$
$$\mu_A(x \in U) \,\&\, (\exists p)(x = (y,p))$$

and extend this definition on other formulas so that:

$$\mu_A(\sim\alpha) = \sim\mu_A(\alpha), \quad \mu_A(\alpha \,\&\, \beta) = \mu_A(\alpha) \,\&\, \mu_A(\beta),$$
$$\mu_A(\alpha \vee \beta) = \mu_A(\alpha) \vee \mu_A(\beta), \quad \mu_A((\exists x)\alpha) = (\exists x)\, \mu_A(\alpha),$$
$$\mu_A((\forall x)\alpha) = (\forall x)\, \mu_A(\alpha)$$

In other words, we define occurrences of elementary situations as the corresponding situations accompanied by intervals in which they maintain, and occurrences of elementary changes as the corresponding elementary changes accompanied by phases at which they take place.

It remains to prove that μ_A carries over all theorems of SEM(A) onto theorems of sem(A). Since μ_A preserves logical operations we may limit ourselves to axioms only.

It is a matter of routine to verify that the axioms (E1)-(E12) are carried over onto theorems of sem(A), and that the corresponding ordering of occurrences of elementary situations is identical with the following:

$t \lessdot t'$ iff there are elementary situation occurrences
$t = t_1 = (x_1, p_1, q_1), \ldots, t' = t_k = (x_k, p_k, q_k)$ or (x_k, p_k),
and elementary change occurrences
$u_1 = (y_1, q_1), \ldots, u_{k-1} = (y_{k-1}, q_{k-1})$, such that
$x_i \in L(y_i)$, $x_{i+1} \in R(y_i)$, $p_{i+1} = q_i + 1$ for $i = 1, \ldots, k-1$.

To prove that (E13) converts into a theorem, suppose that an instance m of a rule of A is possible in the situation $F(c_{max})$ corresponding to the case c_{max} of the maximal occurrences of elementary situations. Suppose that m has no occurrence with all the preconditions in c_{max} and no postcondition. If m is not in a conflict with the changes which have occurrences with the preconditions in c_{max} then no elementary situation from L(m) ceases and no elementary situation from R(m) arises. Hence, there is a phase $q \in P$ such that Possible(m,p) for all $p \geq q$. However, due to (E'5), this is impossible. Thus, m must be in a conflict with changes which occur in the case c_{max}, or m has an occurrence with all the preconditions in c_{max} and no postcondition. In other words, (E13) converts into a theorem.

In consequence, we obtain the following:

<u>Modelling Theorem</u> For every algorithm A the correspondence μ_A is a modelling of the general objective semantics SEM(A) in the general subjective semantics sem(A).

This theorem extends to special semantics if the specific axioms which are added to SEM(A) are appropriately reformulated and added to sem(A). For instance, the axiom

$$(\forall t)(F(t)= R(a,b,c) \Longrightarrow c= a+b)$$

may be reformulated as

$$(\forall p)(R(a,b,c) \in G(p) \Longrightarrow c= a+b)$$

In this way we obtain a positive solution of the stated problem.

6. COMMENTS AND CONCLUSIONS

What we have presented is a formalism for characterizing processes which are studied in computer science.

This formalism is not a language in the strict sense though it could easily be developed to a language. We must also emphasize that it is a tool for describing rather than for programming processes. We do not know, as yet, how to implement it in an efficient way, and convert into a programming language. Besides, it seems to be not very convenient for programming.

Our intention was rather to offer a tool to analyse various processes (especially non-sequential ones) in a mathematical way. Having this in mind we give a formalism universal enough to cover typical processes, together with precisely defined mathematical semantics.

An important result is that objective semantics can be modelled in subjective ones. It justifies in a precise way the approaches which base on replacing parallelism by indeterminism.

The formalism covers notions like Markov algorithms, grammars (including those multidimensional and graph grammars of Ehrig, Pfender, Schneider[2]), various schemes of computations (in this number polyadic ones), and offers semantics of these notions. It is also a tool to define Petri nets (elementary situations and elementary changes are detailed descriptions of what is called conditions and events in Petri[9]).

Received May 20, 1976

REFERENCES

1. Dijkstra,E.W., Hierarchical Ordering of Sequential Processes, Acta Informatica 1, Springer-Verlag 1971
2. Ehrig,H., Pfender,M., Schneider,H.J., Kategorielle Konstruktionen in der Theorie der Graph-Grammatiken, in Fachgespräch über mehrdimensionale formale Sprachen, Erlangen, 1973
3. Genrich,H.J., Einfache nicht-sequentielle Prozesse, BMBW--GMD-37, Bonn, 1971
4. Karp,R.M., Miller,R.E., Parallel Program Schemata, JCSS 3(2)
5. Mazurkiewicz,A., Proving Properties of Processes, CC PAS Reports 134, 1973
6. Mazurkiewicz,A., Parallel Recursive Program Schemes, Proc. of MFCS´75 Symp., Marianske Lazne, Lecture Notes in Comp. Sc. 32, Springer-Verlag, 1975
7. Mazurkiewicz,A., Invariants of Concurrent Programs, to appear
8. Milner,R., An Approach to the Semantics of Parallel Programs, Unpublished Memo, Comp. Sc. Dept., Univ. of Edinburgh, 1973
9. Petri,C.A., Concepts of Net Theory, Proc. of MFCS´73 Symp., High Tatras, 1973
10. Petri,C.A., Interpretations of Net Theory, GMD, Bonn, 1975, presented at MFCS´75 Symp., Marianske Lazne, 1975
11. Winkowski,J., Towards an Understanding of Computer Simulation, CC PAS Reports 243, 1976
12. Winkowski,J., On Sequential Modelling of Parallel Processes, CC PAS Reports 250, 1976
13. Rajlich,V., Relational definition of computer languages, Proc. of MFCS´75 Symp., Marianske Lazne, Lecture Notes in Comp. Sc. 32, Springer-Verlag, 1975