PARTIAL AND TOTAL RELATIVE CORRECTNESS FOR ANALYSIS OF REAL–TIME SYSTEMS

TOMASZ SZMUC*

This paper is a continuation of the former works dealing with the correctness verification of concurrent systems. The correctness in the sense of criterion (relative correctness) is a basic notion in these investigations. Two classes of the problem: partial and total relative correctness are considered. Formal tools and conception of computer support for the correctness verification are developed.

1. Introduction

Relative correctness is a relation between verified process (sequential description of verified system) and criterion process (specification of correctness requirements). In the processes algebra area the introduced partial correctness corresponds to the macro–homomorphism (Szmuc, 1986; 1989a; 1989b) from the verified process to the criterion one, while the total correctness corresponds to the macro–endomorphism. On the other hand, these notions may be treated as generalizations of partial and total correctness which are defined for sequential programs. Moreover, strong analogies between safety and liveness properties (Manna and Pnueli, 1974) may be observed.

2. Partial and Total Relative Correctness

The notation used in this paper was introduced in the former works by the author. For any relation $T \subseteq X \times Y$ the domain (range) of the relation is denoted by Dom T (Ran T). If T is the relation defined above, then for any $x \in X$: $T(x) = \{y \mid (x, y) \in T\}$.

For any subset $A \subseteq X$, the image of the subset is denoted by $T(A) = \bigcup_{x \in A} T(x)$. This operation is extended for the empty set, i.e. : $T(\emptyset) = \emptyset$.

*Institute of Automatics University of Mining and Metallurgy, al. Mickiewicza 30, 30-059 Kraków, Poland

Sequential nondeterministic process (Bartol *et al.*, 1977; Pawlak, 1969) is a generalization of automata notion. This process is the fundamental notion in the corresponding algebra (Szmuc, 1986; 1989a; 1989b), which is a language for the correctness considerations.

Definition 1. By a *process* we mean a relational structure $P = (S, B, F, T)$, where S is a countable set of states, $B \subseteq S$ is a set of initial states, $F \subseteq S$ is a set of final states, $T \subseteq S \times S$ is a transition relation, and the following condition is satisfied:

$B \subseteq \text{Dom}\, T$ and $F \cap \text{Dom}\, T = \emptyset$. ■

Any sequence of $n (n \geq 1)$ states of process P, $sc(s_1, \cdot) = {}^1(s_1, \ldots)$ is called *semicomputation* iff any of the two successive (in the sequence) elements s_i, s_{i+1} are in the relation, $(s_i, s_{i+1}) \in T$. The set of all elements which are in semicomputation sc is denoted by $\varphi(sc)$. Any semicomputation $sc(s_1, \cdot)$ such that $s_1 \in B$ is called *computation* iff the semicomputation ends in a state that belongs to F.

The relative correctness is defined for a verified process (describing concurrent system) and correctness criterion, which consists of criterion process and correctness relation. This relation connects selected (so–called characteristic) states in the verified process with the corresponding states in the criterion process. The correctness may be interpreted as a convergence between verified system (exact description) and its specification (rough model). The correctness definitions will start from the same auxiliary notions as used in (Szmuc, 1989a; 1989b).

Let $P = (S, B, F, T)$, $P' = (S', B', F', T')$ be processes and $k \subseteq S \times S'$ a relation. For every pair $(s, s') \in k$ we define:

– *lower semicomputation* referring to the pair – such a semicomputation $sc(s, \cdot)$ that the following condition is satisfied:

$$(\forall s_1 \in \varphi(sc(s, \cdot))) \quad (\forall s_1' \in k(s_1)) \quad (s', s_1') \notin T';$$

– *upper semicomputation* referring to the pair – such a semicomputation $sc(\cdot, s)$ that the below condition is satisfied:

$$(\forall s_0 \in \varphi(sc(\cdot, s))) \quad (\forall s_0' \in k(s_0)) \quad (s_0', s') \notin T'.$$

A set of lower semicomputations referring to a pair $(s, s') \in k$ is denoted by $SL(s, s')$, while the set of the upper ones by $SU(s, s')$. The generalizations

[1] Symbol"$\cdot$" means any.

referring to subsets $X \subseteq S$ and $Y \subseteq S'$ are defined:

$$SL(X,Y) = \bigcup_{s\in X,\ s'\in Y} SL(s,s') \quad \text{and} \quad SU(X,Y) = \bigcup_{s\in X,\ s'\in Y} SU(s,s').^2$$

We additionally introduce closures of the sets:

$$\overline{SL}(s_0,s') = \{(s_0)\} \cup \{sc(s_0,s_n) \mid sc(s_0,s_{n-1}) \in SL(s_0,s')\};$$

$$\overline{SL}(s_n,s') = \{(s_n)\} \cup \{sc(s_0,s_n) \mid sc(s_1,s_n) \in SU(s_n,s')\}, \quad \text{see [12,13].}$$

The generalizations referring to subsets of the sets, S, S' may be correspondingly transformed to the closures defined above. In the case when argument is a maximal set will be denoted by "*", e.g. $SU(B,^*)$.

Definition 2. Let $P = (S,B,F,T)$, $P' = (S',B',F',T')$ be processes and $k \subseteq S \times S'$ a relation. We say that *process P is partially correct in the sense of the correctness criterion* (P',k) iff the following conditions are satisfied:

1. $SU(B,^*) = SU(B,B')$;
2. $SL(F,^*) = SL(F,F')$;
3. for any $s'_n \in \text{Ran } k \cap \text{Ran } T'$, there exist states s_0, s_n, s'_0 such that $s'_0 \in T'^{-1}(s'_n)$ and the condition is satisfied:
   $\overline{SL}(s_0,s') \cap \overline{SU}(s_n,s'_n) \neq \emptyset$. ■

It is easy to notice, that the above mentioned definition is equivalent to the one, introduced in the former works (Szmuc, 1986; 1988; 1989a; 1989b), however the form seems to be simpler and more clear.

Let us assume that P, P' considered in any correctness problem are finite, and may be specified by sets of computations (Szmuc, 1986; 1989b). The latter property means, that if any computation (in any process) is finite then it ends in the final state of the process.

Let $SC(X,Y)$ be a set of lower or upper semicomputations defined as above. The set of all states of criterion process which are referred to by semicomputations from the set is denoted by $CH(SC(X,Y))$, i.e. :

$$CH(SC(X,Y)) = \{s' \mid (\exists s)\ SC(s,s') \subseteq SC(X,Y)\}.$$

Corollary 1. If process P is partially correct in the sense of (P',k), then:

[2] The unions are defined for all these pairs $(s,s') \in X \times Y$ for which the corresponding semicomputation exists.

1. $CH(SU(B,^*)) \subseteq B'$;
2. $CH(SL(F,^*)) \subseteq F'$;
3. Ran $k \subseteq S'$.

Definition 3. Let $P = (S, B, F, T)$, $P' = (S', B', F', T')$ be processes and $k \subseteq S \times S'$ a relation. We say that *process P is totally correct in the sense of the correctness criterion* (P', k) iff the following conditions are satisfied:

1. $CH(SU(B,^*)) = (B')$;
2. $CH(SL(F,^*)) = (F')$;
3. for any $s'_n \in$ Ran T', there exist states s_0, s_n and s'_0 such that $s'_0 \in T'^{-1}(s'_n)$ and the condition is satisfied:
   $\overline{SL}(s_0, s') \cap \overline{SU}(s_n, s'_n) \neq \emptyset$. ■

Corollary 2. If process P is totally correct in the sense of (P', k), then:

1. $CH(SU(B,^*)) = B'$;
2. $CH(SL(F,^*)) = F'$;
3. Ran $k = S'$.

Informally speaking process P is totally correct in the sense of correctness criterion (P', k) iff k–projection of P into covers process P'.

Example 1. Let us consider process $P = (S, B, F, T)$ and correctness criterion (P', k) defined in the following way:

1. $S' = B' \cup F'$;
2. $T' \subseteq B' \times F'$;
3. Dom $k \subseteq B \cup F$.

It is easy to notice, that the class of partial (relative) correctness problem corresponds to the classical partial correctness notion (Manna, 1974). The interpretation of the total correctness does not meet directly the classical formulation. Let us additionally assume that Dom $k = B \cup F$ and Ran $k = S'$. This relative total correctness includes a requirement of concordance between infinite computations too. It must be mentioned however, that

these computations may be detected using more detailed specification of the criterion, e. g. by adding intermediate characteristic states indicating infinite loops. The general formulation may be a first step in the analysis and may support specification of the intermediate states.

Example 2. Safety property may be defined using temporal logic (Manna, 1981):

$| \equiv b(x) \Rightarrow \Box w$, where $b(x)$ is a precondition of program (process) and w is a formula.

This formula may be transformed into the equivalent specified by "process language" one:

$(\forall i \geq 0)\ w(i)$, where $w(i)$ means that w is true for final segment of computation, i.e. semicomputation derived from the computation after elemination of "i" initial states.

Let us notice that almost all examples of the property are usually described by implication. Antecedent of this implication specifies a state/states in which property defined by consequence should be satisfied. The class of correctness problems may be defined as the partial relative correctness (Szmuc, 1989b).

Example 3. Liveness property defined using temporal logic (Denis, 1974):

$| \equiv \Box(w_1 \Rightarrow <> w_2)$,

may be transformed into the following one, satisfied for every computation:

$(\forall j)(w_1(j) \Rightarrow (\exists i \geq j)\ w_2(i))$.

It is easy to notice that this property may be described by the total relative correctness (compare with (Szmuc, 1989b)).

2. The Correctness Verification

The verification of the relative correctness may be performed in two ways (Szmuc, 1988; 1989a; 1989b) as a so–called static verification and as dynamic correction. The static verification of the partial correctness and a concept of dynamic correction has been presented in (Szmuc, 1988; 1989b). The verification process consists of four stages: construction of an exprssion that specifies verified process, extention and reduction of the expression (elimination of non–characteristic states), and the correctness verification by simulation of the correspondingly reduced coupled process. The three initial stages are based on algebra of processes which is described in (Szmuc, 1989b). An idea of construction of expression is shown in Figure 1. The simulation is carried out by an analysis of "characteristic computations" corresponding to every reduced subexpression. The conditions from theorem

1 are examined in every transition (state) of the computation. If at least one of them is not satisfied, then the verified process is incorrect in the transition/state.

The process is correct in the sense of a given criterion when such transitions are not found. More detail explanations and corresponding algorithms may be found in (Szmuc, 1989a; 1989b).

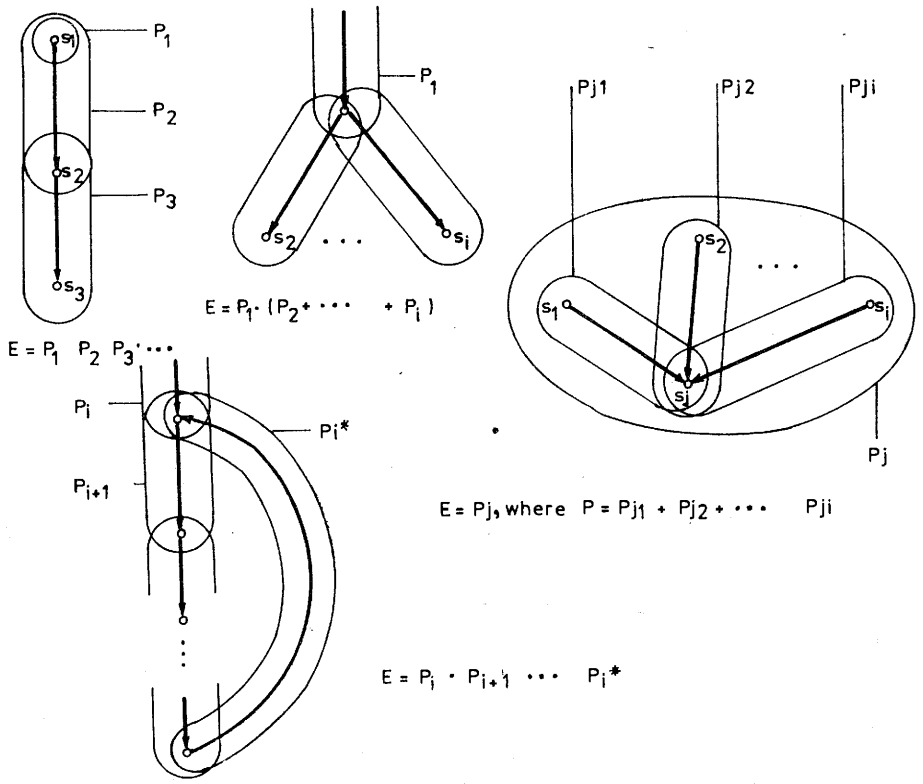

Fig.1. Rules for construction of expression

2.1. Total Correctness Verification

Total correctness verification will be performed in a similar way to the partial one (Szmuc, 1986; 1988; 1989a; 1989b). however the fundamental formal tools: coupled process and local testability theorem will be modified.

Definition 4. Let P, P' be processes and $k \subseteq S \times S'$ a relation. By the *coupled process* we mean a quadruple $\hat{P} = (\hat{S}, \hat{B}, \hat{F}, \hat{T})$, where:

- $\hat{S} \subseteq S \times M$, $M \subseteq 2^{S'} \times 2^{S'} \times 2^{S'}$;

- $\hat{B} = \{(s,(pS',pB',pF')) \mid s \in B \wedge pS' = (B' \cup T'(k(s)) \setminus k(s)) \wedge pB' = B' \cap k(s) \wedge pF' = k(s)\}$;
- $\hat{F} \subseteq \{(s,(\cdot,\cdot,\cdot)) \mid s \in F\}$;
- $\hat{T} \subseteq \hat{S} \times \hat{S}$ – relation such that for any $((s,(pS,'pB',pF')), (s_1,(pS'_1, pB'_1,pF'_1))) \in \hat{T}$ the following conditions are satisfied:
  1. $(s,s_1) \in T$,
  2. $pS'_1 = pS' \cup (T'(k(s_1)) \setminus k(s_1))$,
  3. $pB'_1 = B' \cap k(s_1)$,
  4. $pF'_1 = pF' \cup k(s_1) \setminus (T'^{-1}(k(s_1)) \setminus F'_*)$. ■

The modifications include extention of memory state by adding coordinate (pB') and condition (3) describing changes of the coordinate. The later one contains information about those initial states (of criterion process) which has appeared (in the coupled process) until current state in the verified process have been reached. Set F'_* (condition 4) is a set of pseudo–final states of process P'. Note that any infinite (looping) computation may be represented by a finite sequence of states (finite expression) which describes the corresponding loop (Fig.1). A last element of such a sequence is designated by addition of "*" to a symbol of a first state in the loop (Szmuc, 1989a; 1989b) and Figure 1. It results from condition 4 that if a state from F'_* has appeared in set pF'_* in any state, then the state (with star) will not be removed in future states (of process $\hat{P}$). On the other hand an appearance of a state from F'_* means that all states in the corresponding computation have been reached.

Theorem 2. Let P, P' be processes and $k \subseteq S \times S'$ a relation. Process P is *totally correct in the sense of correctness criterion* (P', k) if for the coupled process $\hat{P}$ (def.4) the following conditions are satisfied:

1. $(\forall s \in B)\ k(s) \subseteq B'$;
2. $(\forall(\cdot,(pS',pB'pF')) \in \hat{F})\ pF' \subseteq F'$;
3. for any transition $((\cdot,pS',\cdot)),(s_1,(\cdot,\cdot))) \in \hat{T}$: $k(s_1) \subseteq pS'$;
4. $\bigcup pB'_{(\cdot,(\cdot,pB',\cdot))\in\hat{F}} = B'$ and $\bigcup pF'_{(\cdot,(\cdot,\cdot,pF'))\in\hat{F}} = F' \cup F'_* \cup \overline{F}'_*$

where $\overline{F}'_*$ is any (in particular empty) subset of states which do not belong to set F'_* and which appear in computations ending in states from F'_*.

Proof. The proof related to conditions 1–3 may be performed in a similar way to the partial correctness in (Szmuc, 1989b). Condition 4 expresses a requirement of convering of process P' by k–projection of process P. The first part of condition 4 means $k(B) = B'$. It results from the second one that $k(F) = F'$ and all states of infinite (looping) computations have appeared ($\bigcup pF' \supseteq F'_*$). Hence it results that all states of process P' belonging to finite and infinite computations have been reached. The fact that set $\overline{F}'_*$ is nonempty means that its elements have appeared more than one time. ■

Let us notice that local (dynamic) conditions related to the partial correctness are complemented by the global condition (4) referring to final states (of process $\hat{P}$). The total correctness verification is rather static in character and for this reason the dynamic correction of the correctness is not considered. It should be noticed however, that the global feature covers only one set of final states (of $\hat{P}$) and is related to the two cordinates only. The fact simplifies the verification process.

2.3. Static Verification of Concurrent Processes

The verified process may be interpreted as a sequential description of a system of concurrent processes. This system consists of a family of the so-called component processes $\{P_n\}_{n\in N}$, a set of states of the system.

Definition 5. *System of (concurrent) processes* is the structure, $I = (\{P_n\}_{n\in N}, S, \Phi)$, where:

- $\{P_n\}_{n\in N}$ – family of processes;
- $S \subseteq X_{n\in N} S_n$ – set of states of the system;
- $\Phi \subseteq \{\phi_{n,m} | n \neq m \wedge \phi_{n,m} \subseteq (S_n \times S_n \times S_m) \times S_m\}$.

Any relation $\phi_{n,m}$ defines an interaction between processes P_n and P_m. For example, $((s_n, s'_n, s_m), s'_m) \in \phi_{n,m}$ means that when in process P_n transition from s_n to s'_n is performed, then this transition causes a simultaneous change from s_m into s'_m in process P_m. Interpretation and a more detailed description may be found in (Szmuc, 1989b). This formal model covers any of the three main types of parallel program constructions (Szmuc, 1989b):

- processes executing within global environment (Brinch, 1976) (shared data, monitors),
- processes communicating by message passing (Communicating Sequential Processes (Hoare, 1985)),

- data flow programs (Denis, 1974).

Behaviour of the system is described by sequential process (Szmuc, 1989b). The construction realizes a transition between the non–sequential (system of processes) and sequential (process describing behaviour of the system) ways of describing parallelism.

Let $I = (\{P_n\}_{n\in N}, S, \Phi)$ be system of processes. A set of all projections from system states into component processes states will be denotes by $H = \{H_n : S \longrightarrow S_n | n \in N\}$.

1. For every $n \in N$ we define *relation* $T^n \subseteq S \times S$ such that:

   $h_n(\mathrm{Dom}\, T^n) = \mathrm{Dom}\, T_n$ and for any $(s, s') \in T^n$:

   a) $(h_n(s), h_n(s')) \in T_n$,

   b) for any $m \in N \setminus \{n\}$:

   $h_m(s') \in \phi_{n,m}(h_n(s), h_n(s'), h_m(s))$ if $(h_n(s), h_n(s'), h_m(s)) \in \mathrm{Dom}\, \phi_{n,m}$

   $(h_m(s') = h_m(s)$ otherwise;

2. For any sequence of different indexes $p = n_1, n_2, \ldots$ we define *relation* $T^{(p)} \subseteq S \times S$:

$$T^{(p)} = \begin{cases} T^{n_1} & \text{if } p = n_1, \\ T^{n_1} \diamond T^{(p_2)} & \text{if } p = n_1, n_2, \ldots \text{ and } p_1 = n_2, \ldots \end{cases}$$

   ($\diamond$ specifies composition of relations).

3. $P = (S, B, F, T)$ is *process describing behaviour of system*

   $I = (\{P_n\}_{n\in N}, S, \Phi)$ when relation T is defined as follows:

   $(s, s') \in T$ iff $(\exists p)\ (s, s') \in T^{(p)}$. ■

Concurrent execution of component processes is simulted by the *interleaving* of their atomic actions. More detailed specification of sets B and F depends on the application, in particular the set F may be empty. The definition is applied in algorithm that translates parallel specification into the equivalent sequential one (Szmuc, 1989b).

The class of correctness problems for verified processes specified by a system of concurrent processes and relation k, which satisfies condition $\mathrm{Dom}\, k \subseteq \bigcup_{n\in N} S_n$ – is very important in the correctness verification. This

property simplifies the three initial stages of verification and increases the level of automation.

Let us consider a system in which every component process is specified using a programming language. Let us additionally assume that characteristic states are specified by selected instructions of the multiprocess program (system). Every component process may be described by the quotient process (Szmuc, 1989a and 1989b). The construction of the expressions specifying the quotient processes may be performed automatically in the compilation (parsing) time. The selected rules of transformations are presented below.

1. Composed instruction, $P_1; P_2; ...; P_n; : E = P_1 \cdot P_2 \cdot ... \cdot P_n;$

2. *while* instruction, *while* C *do* $P_p; P_f; :$

$$E = \begin{cases} P_p + P_p \cdot P_f + P_f & \text{if } P_p \text{ is simple instruction,} \\ P_1 \cdot ... \cdot P_n \cdot P_1^* + P_1 \cdot ... \cdot P_n \cdot P_f + P_f & \text{if } P_p = P_1 \cdot ... \cdot P_n; \end{cases}$$

3. *if* instruction, *if* C *then* P_1 *else* $P_2; : E = P_1 + P_2.$

The above mentioned transformations (and additional in (Szmuc, 1989b)) may be used for the automatic generation of expression specifying component processes.

By theorem 7.3 in (Szmuc, 1989b), the reduced verified process (of the system) may be obtained by a separate reduction in every component process and by constructing this verified process from the reduced component ones. Hence the three initial stages of the verification may be simplified and may be carried out automatically.

The above mentioned class of correctness problems refers to computer network protocols (Szmuc, 1989a) and real–time programs (Szmuc, 1988; Szmuc, 1989a).

2.4. Implementation Remarks

Implementation of the system for relative correctness verification has been developed in the Institute of Automatics for a few years. The system is divided into two layers:

- a kernel implementing the correctness verification (partial and total) for processes described sequentially,
- a collection of modules which realize transformations from different parallel specifications into the sequential one.

The implementation of the kernel has been completed, while realization of the modules is under development. The two modules applying Petri nets specification and CSP like languages seem to be near completion.

2.5. Dynamic Correction

Static verification is based on the reduced description of the verified process and simulation of the corresponding coupled process. Another approach to the verification consists in adding the correction module to the verified process running in the computer system. This process and the module constitute the coupled process. The local testability theorem is used during any computation in the verified process P. Transitions in non–characteristic states are performed without any interference with the module. In every characteristic state process P calls the module, in which conditions of the theorem are examined. If they are satisfied, then the transition may be realized in process P, otherwise the transition must be blocked. The correction module consists of the memory (definition 4) and procedure implementing transition to the next states in coupled process (related to the memory).

Implementation of dynamic correction is simplified in the cases of the correctness problems mentioned in the previous subsection. Blocking of the incorrect transitions is performed thereby by suspensions of the corresponding component processes. Implementations of the correction module in procedural and message–passing systems are discussed in (Szmuc, 1989b).

3. Concluding Remarks

Formal tools and a conception of computer support for specification and the correctness analysis have been presented. The investigations are based on two statements: nondeterministic process, being a fundamental notion, and the relative correctness (partial or total), being defined as a relation between the verified and criterion processes.

The first statement facilitates the description of many objects using the same process notion. Differences are obtained by changing the interpretation of objects constituting the process. The formal tools (homomorphism, congruences, decomposition) may be used in the correctness analysis (Szmuc, 1989a). The general form of the model causes that it may be applied in the development of other formalisms. The correctness is partialy similar to the bisimulation notion (Castellani, 1985) based on Milner's CCS (Milner, 1980). It seems, however, that the correctness enables stronger reduction of specification in the case of parallel systems, because the correctness relation

absorbs the whole permutations of states sequences, which are specified as parallel ones. Hence, the criterion process may be described in a simpler and more natural form. More detail characterization and application of the proposed approach may be found in (Szmuc, 1989a).

References

Bartol W., Raś Z. and Skowron A. (1977): *Theory of computing systems.* In Mazurkiewicz A., Pawlak Z. (eds.): Mathematical Foundations of Computer Science.– Warsaw: PWN–Polish Scientific Publishers, pp.101–165.

Brinch P. (1976): *The SOLO operating systems: a concurrent Pascal program; Processes, monitors; Job interface.*– Software Practice and Experience, v.6, pp.141–149; pp.151–164; pp.165–200.

Castellani M. (1985): *Bisimulation and abstraction homomorphisms.*– Lecture Notes in Computer Science, v.185, pp.223–238.

Denis J.B. (1974): *First version of data flow procedure language.*– Techn. Memo. 61. Lab. for Comp. Sci., MIT, Cambridge Mass.; Also in: Lecture Notes in Computer Science, v.19, pp.362–376.

Hoare C.A.R. (1985): *Communicating Sequential Processes.*– New York: Prentice Hall International.

Manna Z. and Pnueli A. (1974): *Axiomatic approach to total correctness of programs.*– Acta Informatica, v.3, pp.243–263.

Manna Z. and Pnueli A. (1981): *Verification of concurrent programs: The temporal framework*; In: Boyer R.S., Moore J.S. (eds.): The Correctness Problem in Computer Science.– New York: Academic Press, pp.215–273.

Milner R. (1980): *A Calculus of Communicating Systems.*– Lecture Notes in Computer Science, v.92.

Pawlak (1969): *Programmed machines.*– Algorithms, v.10, pp.5–19.

Szmuc T. (1986): *Correctness verification of parallel control programs.* In: Haase H., Knuth E. (eds.): Proc. of the 4th IFAC/IFIP Symp. on Software for Computer Control–SOCOCO'86, pp.159–163.

Szmuc T. (1988): *Correctness verification of real–time programs.*– Prep. of the 15th IFAC/IFIP Workshop on Real–Time Programming, pp.3–9.

Szmuc T. (1989a): *Correctness verification of concurrent systems.*– Proc. of the 22nd Hawaii Inter. Conf. on System Sciences, IEEE Computer Society, v.2, Software Track, pp.295–304.

Szmuc T. (1989b): *Correctness of concurrent software systems.*– Scientific Bulletnes of Academy of Mining and Metallurgy, Automatics, v.46, (in Polish).