ON AXIOMATIC DEFINITION OF MAX-MODEL OF CONCURRENCY

Grazyna Mirkowska[1], Andrzej Salwicki[2]

Abstract

The thesis we present in this paper states that for every concurrent program π there exists a set of modal formulas, also called the axioms $Ax(\pi)$, such that a) the structure of admissible parallel executions of the program π is a Kripke model of the set $Ax(\pi)$ and, b) any Kripke model of the axioms $Ax(\pi)$ is an extension of the structure of all admissible distributed (i.e. parallel) executions of the program π.

INTRODUCTION

The paper presents an attempt to give axiomatic definition of the semantics of concurrent programs. We consider two model of concurrency called MAX semantics (cf.[SM]) and ARB semantics. While the second is based on arbitrary choices of nonconflict instructions, the first requires maximal engagement of processors. We describe a general method which allows to construct axioms A for a given program M with the following properties. 1) The set of all possible computations of the program M in MAX semantics create a model of A called computational model. 2) Every model of A can be restricted to a computational model. Hence in some sense the set A categorically characterizes MAX behaviour of the program M.

We begin our paper with an example illustrating the notions and methods. We present for one program two sets of modal formulas called axioms. The meaning of them is a puzzle for the reader. One set of formulas describes the computations of the program in terms of arbitrary interleavings of atomic actions, second set of axioms

[1] Polish Academy of Sciences
ZSAK, Gliwice

[2] University of Warsaw
Institute of Informatics

describes computations in the environment of maximally involved processors. Then we present the Kripke model for these sets and finally we connect the formulas with a concurrent program. The detailed presentation of the example helps to conceive the differences between two models of concurrent computations.

Sections 2 and 3 are devoted to the definition of MAX semantics and diagrams of programs. In section 4 we generalize method presented in section 1. Various sets of axioms defining local and global behaviour of an arbitrary program in MAX semantics are discussed there. Finally sections 5 and 6 presents the proofs of the main results of the paper.

We assume the reader is familiar with the language of modal logic and the notion of semantic Kripke model for sets of modal formulas. In an Appendix we supply the short resume of these notions.

1. AN EXERCISE IN MODAL LOGIC

In this section we begin with certain sets of modal formulas and we show that they describe semantic behaviour of a concurrent program. The formulas in our first set are constructed from propositional variables $p_1,\dots,p_7,q,q_1,q_2,q_3,q_4$ by means of classical propositional connectives and modality signs: $\Box$ (necessity sign) and $\Diamond$ (possibility sign).

We shall consider the following set A_1 of modal formulas

$$p_1 \Rightarrow \neg (p_2 \vee \dots \vee p_7 \vee q \vee q_1 \vee \dots \vee q_4)$$

$$p_1 \Rightarrow \Box(p_1 \vee \neg p_1 \wedge q)$$

$$q \Rightarrow \Box(p_2 \wedge p_3 \wedge \neg p_1)$$

$$p_2 \Rightarrow \Diamond(q_1 \wedge \neg p_2) \wedge \Diamond p_2$$

$$p_3 \Rightarrow \Diamond(q_2 \wedge \neg p_3) \wedge \Diamond p_3$$

$$q_1 \Rightarrow \Diamond(\neg q_1 \wedge p_5) \wedge \Diamond q_1$$

$$q_2 \Rightarrow \Diamond(\neg q_2 \wedge p_4) \wedge \Diamond q_2$$

$$p_4 \Rightarrow \Diamond(q_3 \wedge \neg p_4) \wedge \Diamond p_4$$

$$q_3 \Rightarrow \Diamond(\neg q_3 \wedge p_6) \wedge \Diamond q_3$$

$$(p_5 \wedge \neg p_6) \Rightarrow \Box p_5$$

$(p_6 \wedge \neg p_5) \Rightarrow \Box p_6$

$(p_5 \wedge p_6) \Rightarrow \Box q_4$

$q_4 \Rightarrow \Box p_7$

$\neg(q_1 \wedge q_3)$

Let us guess what is the semantic meaning of the axioms written above. First, we shall construct a Kripke model for the set A_1 of axioms. The result can be seen on the following diagram, cf. Fig.1.1 . The states of the model are identified by the formulas valid in them, the arrows represent the reachable relation.

Consider the following diagram of a bipartite graph, cf. Fig.1.2. The circles and rectangles form its set of vertices. Each arc is either an arrow from a circle to a rectangle or an arrow from a rectangle to a circle. The circles are denoted by $p_1,\ldots,p_7$, the rectangles are denoted by $q,q_1,\ldots,q_4$. Now, let us play a game of tokens. The rules are defined by the set of axioms. For example, if there is a token in the circle p_2 then it is acceptable (possible) that it stays in place and it is also possible that it is moved into the rectangle q_1. The game begins with one token in the circle p_1 and is over when a token reaches the circle p_7. Observe that the Kripke model of the set A_1 brings the structure of all possible moves in our game. It is not necessary to stress out the similarities between our example and the nets. In the sequel we are extending our example.

Now, let us change the set of axioms. Let the signs α, β, γ denote formulas. We are introducing schemes of axioms with the subformulas $\alpha(x/1)$, $\alpha(x/a)$, $\beta(x/0)$, $\beta(x/b)$, $\gamma(x/b)$, $\gamma(x/a)$ within three schemes below. It means that there are infinitely many axioms and that they are not necessarily of propositional modal logic, they are formulas of first order modal logic.

The set A_2

$p_1 \Rightarrow \neg (p_2 \vee \ldots \vee p_7 \vee q \vee q_1 \vee \ldots \vee q_4)$

$p_1 \Rightarrow \Box(p_1 \vee \neg p_1 \wedge q)$

$q \Rightarrow \Box(p_2 \wedge p_3 \wedge \neg p_1)$

$p_2 \Rightarrow \Diamond(q_1 \wedge \neg p_2) \wedge \Diamond p_2$

$p_3 \Rightarrow \Diamond(q_2 \wedge \neg p_3) \wedge \Diamond p_3$

$(q_1 \wedge \alpha(x/1)) \Rightarrow \Diamond(\alpha(x/a) \wedge \neg q_1 \wedge p_5) \wedge \Diamond q_1$

$(q_2 \wedge \beta(x/0)) \Rightarrow \Diamond(\beta(x/b) \wedge \neg q_2 \wedge p_4) \wedge \Diamond q_2$

$p_4 \Rightarrow \Diamond(q_3 \wedge \neg p_4) \wedge \Diamond p_4$

$(q_3 \wedge \gamma(x/b)) \Rightarrow \Diamond(\gamma(x/a) \wedge \neg q_3 \wedge p_6) \wedge \Diamond q_3$

$(p_5 \wedge \neg p_6) \Rightarrow \Box p_5$

$(p_6 \wedge \neg p_5) \Rightarrow \Box p_6$

$(p_5 \wedge p_6) \Rightarrow \Box q_4$

$q_4 \Rightarrow \Box p_7$

$\neg(q_1 \wedge q_3)$

The meaning of a formula $(\alpha(x/1) \Rightarrow \Diamond\alpha(x/a))$ is as follows: if a state at the beginning of an arrow satisfies $\alpha(x/1)$ then the state at the end of the arrow satisfies $\alpha(x/a)$. Since it is true for every formula $\alpha(x)$, it says that the value of the variable a at the end of the arrow is 1. Compare this axiom with the corresponding axioms of assignment instructions in algorithmic, dynamic or other logic of programs, cf.[MS].

With these explanations we are able to verify that the diagram presented in Fig.1.3 is the model of the set A_2 of axioms. Moreover it describes behaviour of the net in Fig.1.4. Observe that there are two terminal states, i.e. the states with no arrow leaving them. The value of the variable a is 1 in one terminal state and 0 in another. We can verify that the formula

$$(p_1 \Rightarrow \Diamond^8(p_7 \wedge a=1)) \wedge (p_1 \Rightarrow \Diamond^7(p_7 \wedge a=0))$$

is valid in the model. Really, there are two paths starting in the initial state of the Kripke structure and leading in 8 (or respectively in 7) steps to terminal states. In one of the states the value if the variable a equals 1, in another state value of a is 0.

Now, we shall add a new formula to the set A_2

(Max) $\qquad (p_2 \wedge p_3) \Rightarrow \Box(q_1 \wedge q_2 \wedge \neg p_2 \wedge \neg p_3)$

The model for the set $A_2 \cup$(Max) exists and is easy to construct, cf.Fig.1.5. Simply, one has to reject the states reachable from the state $(p_2 \wedge p_3)$ other than $(q_1 \wedge q_2)$.

Observe that in the new model the following formula holds

$$p_1 \Rightarrow \Box^7(p_7 \wedge a=0).$$

Certain behaviours are excluded, and therefore is necessary that after 7 steps we land in a state described by the condition $(p_7 \wedge a{=}0)$.

Let us look closer at the formula Max. It states that whenever two processes are ready to execute their atomic actions q_1 and q_2 (the readiness has place iff the formula $(p_2 \wedge p_3)$ is satisfied) then eventual agents **must** involve themselves in the execution of the actions q_1 and q_2. It is possible since the actions are non-conflict ones.

Till now we have seen three sets of axioms, three models for these sets and two diagrams: one is a concurrent program and another represents its, simple, control structure. As many others authors we came to the conclusion that a discussion of behaviours of a concurrent program requires additional information about the (distributed) state of control. Hence we introduce additional variables. We shall call them control variables. We distinguish variables which assume the value **true** when a process of a concurrent program is ready to execute an action and variables which have the value **true** when a process is executing its atomic action.
Let us consider now another example with iteration (see the diagram in Fig.1.6). The program M reads as follows:

M: **cobegin while** r **do** $x{:=}x{+}1$ **od** $\parallel$ $r{:=}$**false coend**

The following set A_3 of formulas describes the structure of computations of the program:

{local transitions}

$p_1 \Rightarrow \Box q_1$

$q_1 \Rightarrow \Box(p_2 \wedge p_3 \wedge \neg p_1)$

$p_2 \Rightarrow \Diamond(q_2 \wedge \neg p_2) \wedge \Diamond p_2$

$p_3 \Rightarrow \Diamond(q_3 \wedge \neg p_3) \wedge \Diamond p_3$

$(r \wedge q_2) \Rightarrow \Diamond(p_4 \wedge \neg q_2) \wedge \Diamond q_2$

$(\neg r \wedge q_2) \Rightarrow \Diamond(p_5 \wedge \neg q_2) \wedge \Diamond q_2$

$(\alpha(x/\mathbf{false}) \wedge q_3) \Rightarrow \Diamond(\alpha(x/r) \wedge \neg q_3 \wedge p_6) \wedge \Diamond q_3$

$p_4 \Rightarrow \Diamond p_4 \wedge \Diamond(\neg p_4 \wedge q_4)$

$(B(y/x{+}1) \wedge q_4) \Rightarrow \Diamond q_4 \wedge \Diamond(B(y/x) \wedge \neg q_4 \wedge p_2)$

$(p_5 \wedge \neg p_6) \Rightarrow \Diamond p_5$

$(p_6 \wedge \neg p_5) \Rightarrow \Diamond p_6$

$(p_5 \wedge p_6) \Rightarrow \Box q_5$

$q_5 \Rightarrow \Box p_7$

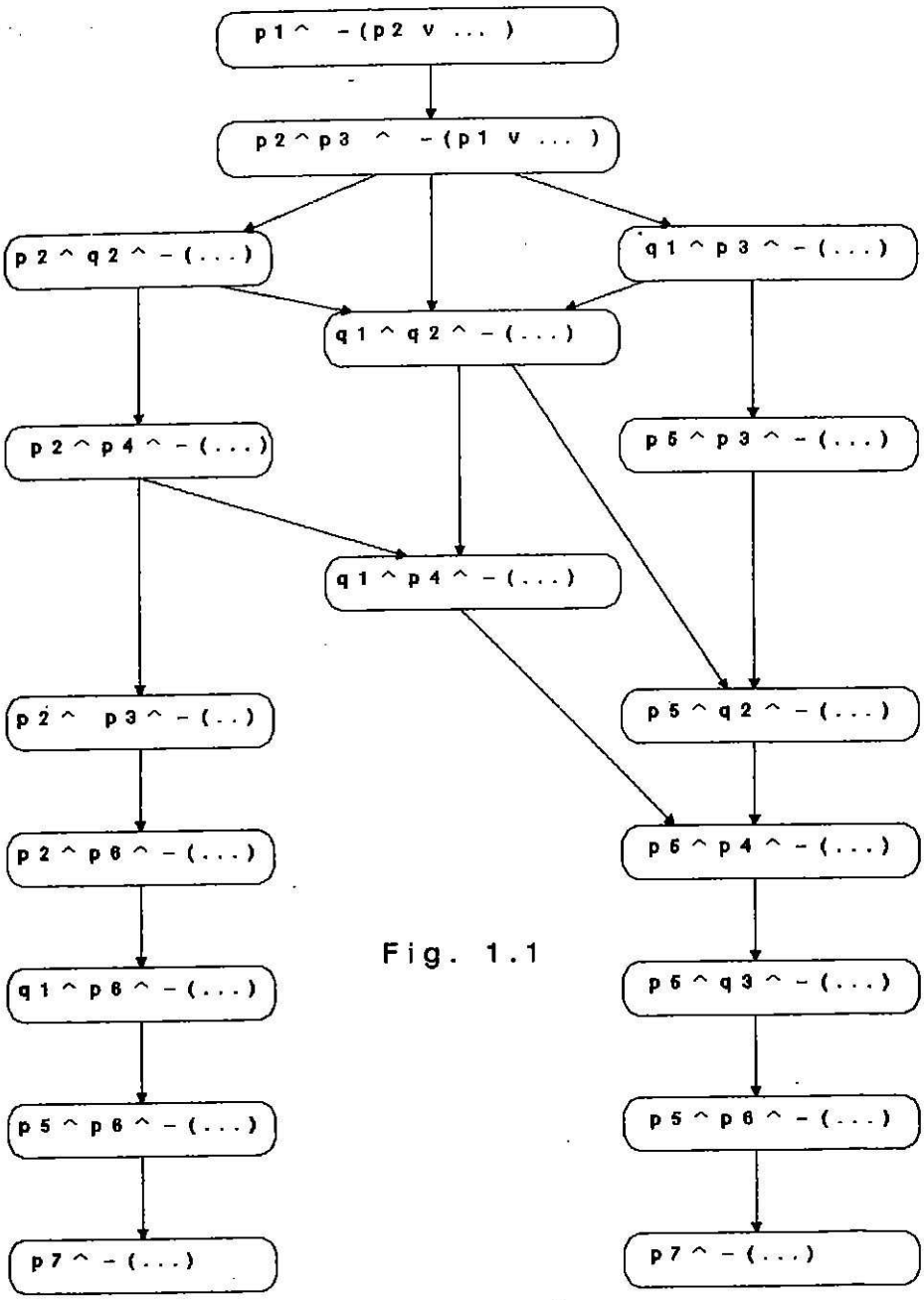


Fig. 1.1

Full Kripke model for the set A1

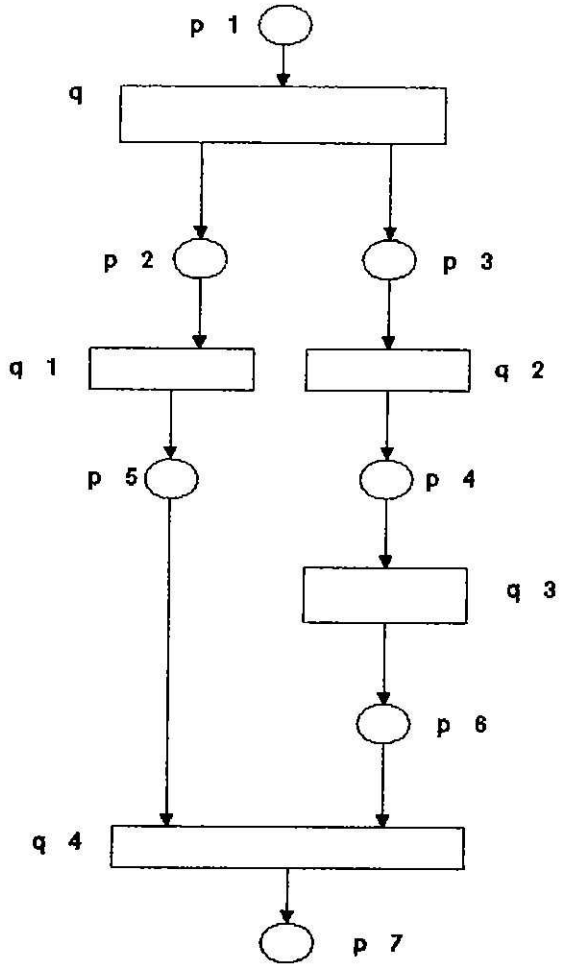


Fig. 1.2

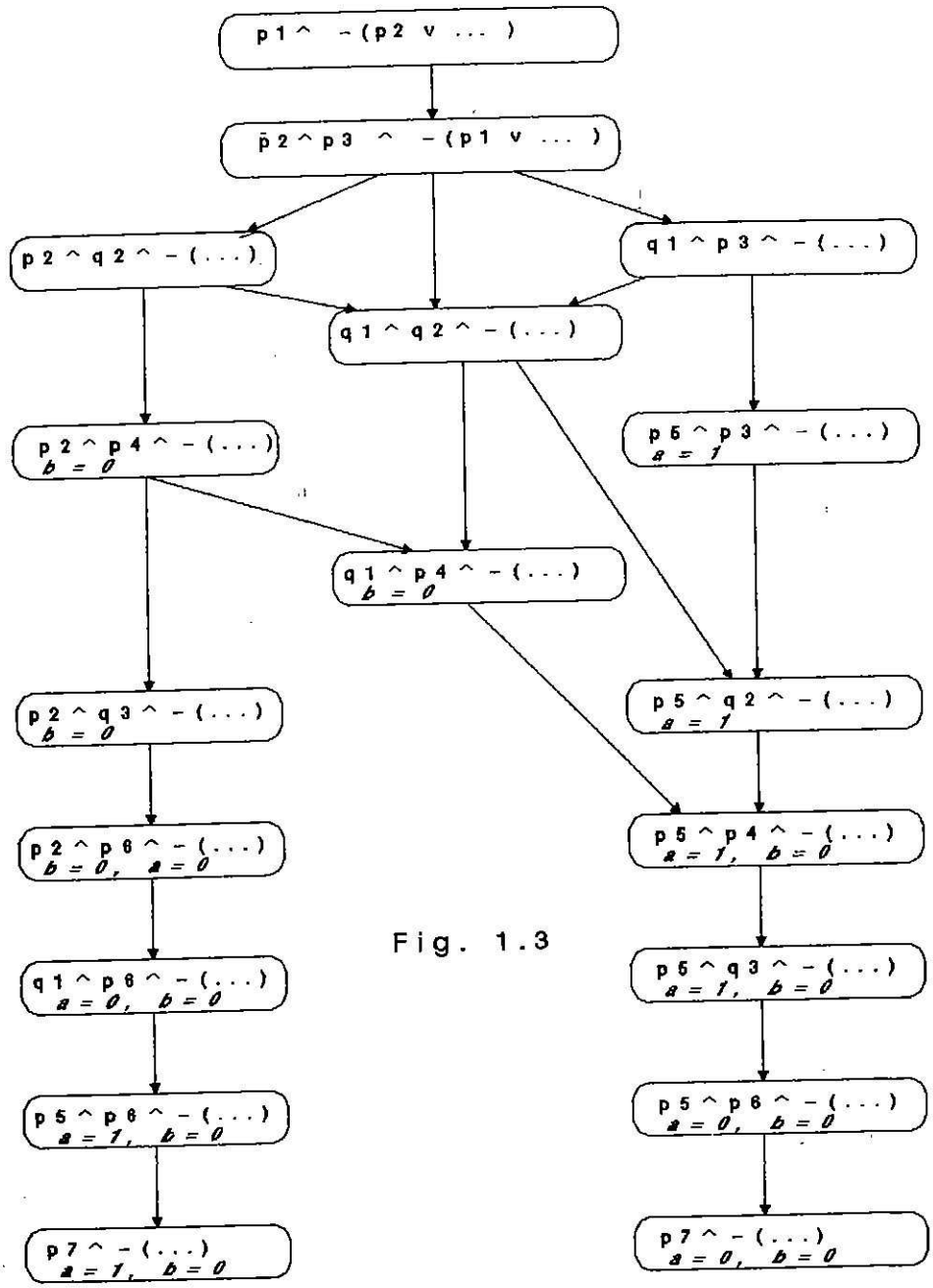


Fig. 1.3

Full Kripke model for the set A2

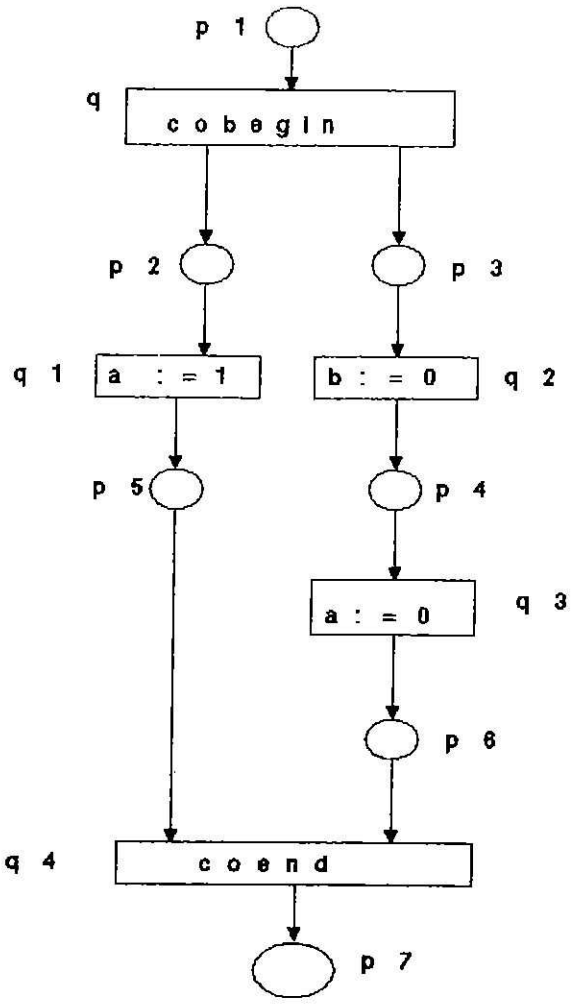


Fig 1.4

cobegin a: = 1 b: = 0; a: = 0 coend

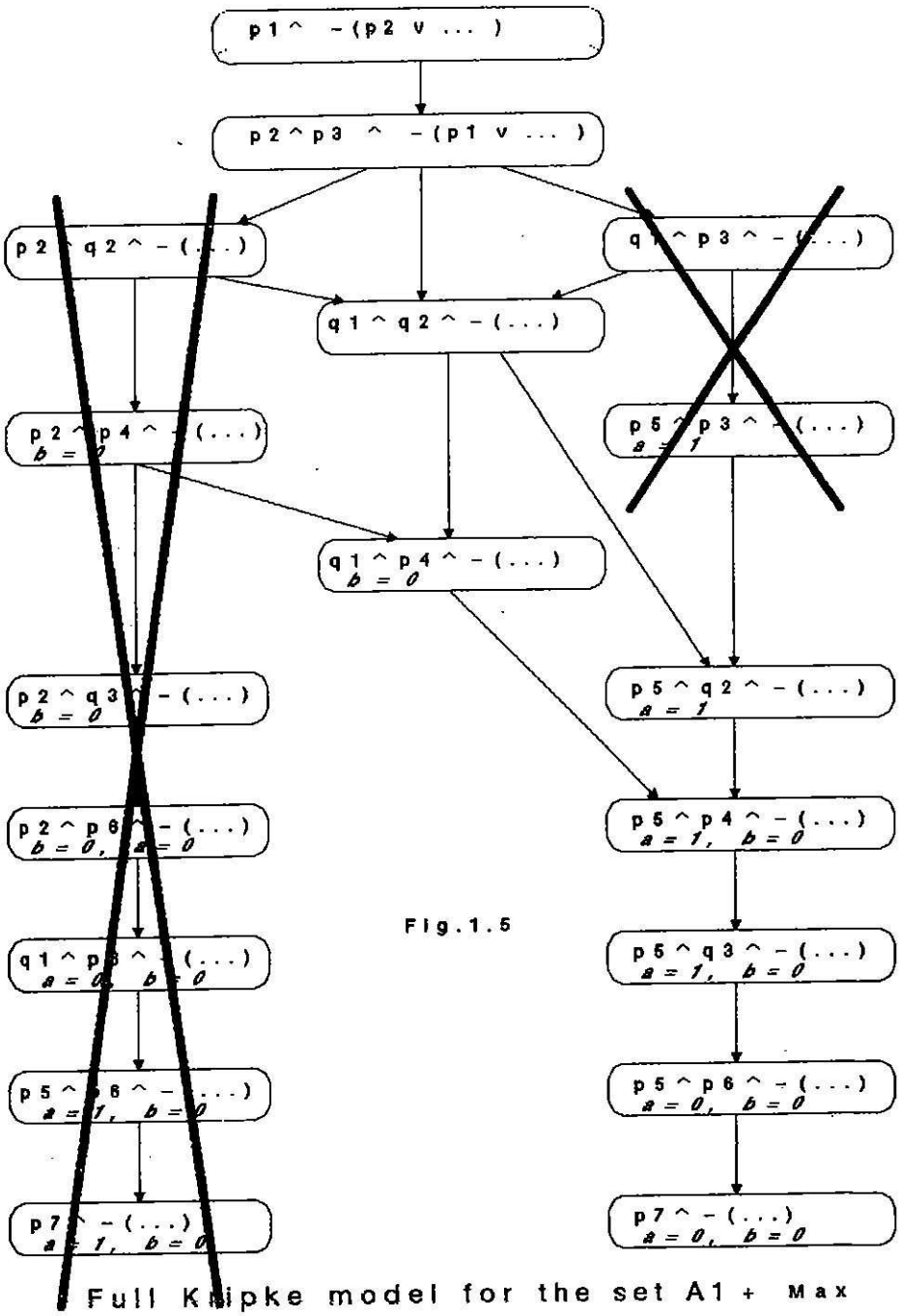
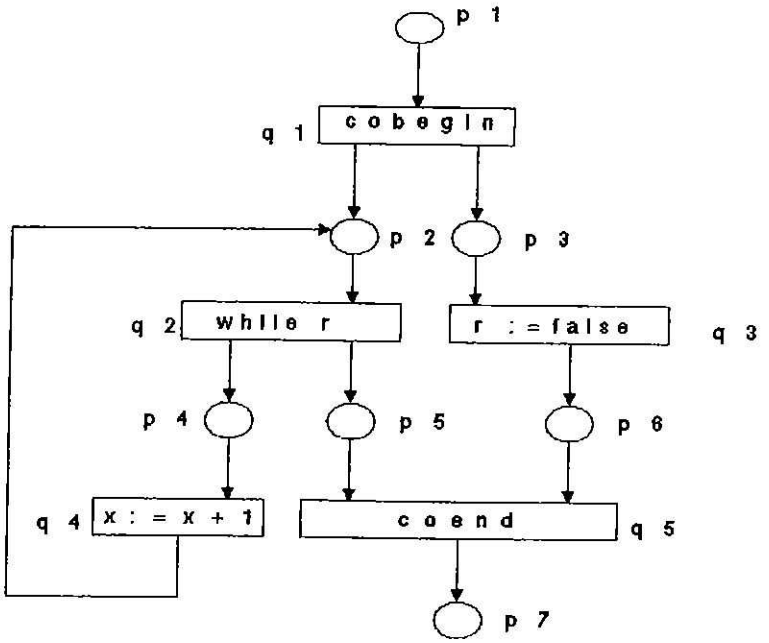


Fig.1.5

Full Kripke model for the set A1 + Max



cobegin while r do x:+x+1 od // r:=false coend

Fig. 1.6

{conflict}

$\neg(q_1 \wedge q_2)$

{initial state}

$p_1 \Rightarrow \neg(p_2 \vee p_3 \vee p_4 \vee p_5 \vee p_6 \vee p_7 \vee q_1 \vee q_2 \vee q_3 \vee q_4 \vee q_5)$

{axiom of maximal concurrency}

$(p_4 \wedge p_3) \Rightarrow \Box(q_4 \wedge q_3 \wedge \neg p_4 \wedge \neg p_3)$

There are two problems connected with the question of existence of a model for the above set of axioms:

First, the structure of a model will be infinite. Should we forget about the formula β, a finite model can be constructed. When the full content of axioms is taken into account, we shall realize that the structure of any model must contain paths of any finite length and by Koenig's lemma it must contain an infinite path.

Second, when one recalls non-standard models of the arithmetic of natural numbers, then it is immediately seen that there exist non-standard models of the the set A_3 of above axioms as well. It means that an additional specification is required. One can demand that the model should be the least model of axioms. This would be a meta-axiom or rule. Another person can add an explicit axiom stating this. We believe that algorithmic methods can be used here. On the other side we have the problem of the admissibility of the principle of maximal engagement. It turns out that in MAX semantics the program we are studying has only finite computations.

In ARB semantics one may observe computations of arbitrarily many steps and also infinite computations. We can state that the formula $\neg\Box M\mathbf{true}$ holds. The meaning of the formula is there exists a nonterminating computation of the program M.

In MAX semantics we observe not only termination property of the program M but moreover the fairness property of M

$$(\exists n)\ ((x=K \wedge r) \Rightarrow \Box M(x < K+n)).$$

The more detailed study shows that the value of n is less than 2

2.SYNTAX AND SEMANTICS

Our aim is to convince the reader that the remarks made in the preceding section have more general character. In this section we shall present a simple language of concurrent programs and its se-

mantics. Later we shall formulate a thesis about axiomatizability of semantics of concurrent programs.

Let L be a first order language with certain set of functional and relational symbols. We shall consider the class of **while** programs extended by the construction **cobegin coend** over the same set of functional and relational symbols.

DEFINITION
The class π of concurrent programs is the least set of expressions which contains all assignment instructions of the form

$$x:=\tau \quad \text{or} \quad q:=y$$

where τ is a term, y is an open formula, x is an individual variable and q is a propositional variable, and such that the class π is closed under the following formation rules: if expressions $M_1,\dots,M_n$ are in π, if y is an open formula in L then the expressions **if** y **then** M_1 **else** M_2 **fi**, $M_1;M_2$, **while** y **do** M_1 **od**, **cobegin** $M_1 \| \dots \| M_n$ **coend** are in π. □

The programs $M_1,\dots,M_n$ in a **cobegin coend** statement will be called processes. Before we present a strict semantics of concurrent programs let us stress that processes are arbitrary programs. In particular, it means that the sets of variables $V(M_i)$ occurring in processes are not necessarily disjoint ones. It means that certain actions of processes are in conflict and that they could not be executed in parallel.

DEFINITION
We shall say that two instructions K and M are in conflict iff K is an assignment instruction $z:=\tau$ and M contains the same variable z, i.e. it is in one of the following forms:

$x:=\omega$, **if** $\gamma(x)$ **then** ...**fi**, **while** $\gamma(x)$ **do**...**od**, $y:=\omega'(x)$.

A set of instructions is a conflict set iff it contains a pair of conflict instructions. □

EXAMPLE
Consider two sets J_1 and J_2 of instructions:

$J_1 = \{\ x:=y+z,$ **while** $x>0$ **do** $y:=\tau$ **od** $\}$

$J_2 = \{\ x:=y+z,$ **while** $y>0$ **do** $x:=\tau$ **od** $\}$

J_1 is a conflict set while J_2 is a non-conflict set. □

DEFINITION
We shall say that a set I is a maximal non-conflict subset of a set J of instructions iff I is a non-conflict set and for every set I' if $I \subset I' \subset J$ and $I' \neq I$ implies that I' is a conflict set.

□

Let us consider an arbitrary data structure A for the language L. By a configuration in A we mean a pair $\langle v,M\rangle$, where v is a valuation in A and M is a program in which some instructions are marked by $*$ or by $\circ$. An instruction K is marked by $\circ$ when $\circ$ occurs just in front of K or when K is of the for **cobegin** $\circ$ **coend**.An instruction K is marked by $*$ when $*$ occurs in front of K or when K is of the form **cobegin** $*\|*\|\cdots\|*$ **coend**.

Below we describe the successorship relation $\xrightarrow{max}$ in the set of all configurations.

DEFINITION
Let $\langle v,M\rangle$ be a fixed configuration and let IN be the set of all occurrences of instructions in M marked by circle $\circ$ and AT be the set of all occurrences of instructions in M marked by $*$.

(2.1) If IN is a maximal in IN∪AT nonconflict set then for arbitrary $J \subset$ IN the configuration $\langle J(v),J(M)\rangle$ is a successor of $\langle v,M\rangle$ (in symbols $\langle v,M\rangle \xrightarrow{max} \langle J(v),J(M)\rangle$) where $J(v)$ is a result of execution of all assignment instructions from J at the valuation v and $J(M)$ is a result of simultaneous replacement of all instructions from J (with marks and separators if necessary) by its reducts according to the following rules of reductions.

$\circ$**if** γ **then** M_1 **else** M_2 **fi** ---> $*M_1$ if $A,v \models \gamma$
$\circ$**if** γ **then** M_1 **else** M_2 **fi** ---> $*M_2$ if $A,v \models \neg\gamma$
$\circ$**while** γ **do** M **od** ---> $*M$; **while** γ **do** M **od** if $A,v \models \gamma$

$\circ$**while** γ **do** M **od** (;) ---> $*$ if $A,v \models \neg\gamma$
$\circ(x:=\tau)$ (;) ---> $*$
$\circ$**cobegin** $*$ **coend** (;) ---> $*$ (with separator ; if occurs)
$\circ$**cobegin** $M_1\|\cdots\|M_n$ **coend** ---> **cobegin** $*M_1\|\cdots\|*M_n$ **coend**

(2.2) If IN is not a maximal nonconflict set then for arbitrary set $J \subset$ AT, such that $J \cup$ IN is a maximal nonconflict set, the configuration $\langle v,M'\rangle$ is a successor of $\langle v,M\rangle$, i.e.,

$$\langle v,M\rangle \xrightarrow{max} \langle v,M'\rangle$$

whenever M' is obtained from M by replacing marks of all instructions indicated by J to $\circ$ according to the following rules

cobegin $\sharp\|\sharp\|\cdots\|\sharp$ **coend** --> **cobegin** $\circ$ **coend**

$\sharp K$ --> $\circ K$ for all other instruction K. □

Removing word "maximal" from the above definition we obtain the other kind of semantics called ARB-semantics (since the set J will denote now arbitrary non-conflict set). The notion of successor obtained in this way we shall denote by $\overset{arb}{\rightarrow}$. Obviously it is not necessary to use two kinds of marks when talking about ARB-semantics.

DEFINITION

By a computation in MAX-semantics (or in ARB-semantics) of the program M in **A** and v we understand any maximal chain of configurations, in the sense of relation $\overset{max}{\rightarrow}$ (or relation $\overset{arb}{\rightarrow}$), with the initial configuration of the form $\langle v, \sharp M\rangle$. □

3. DIAGRAMS OF PROGRAMS

Let M be a fixed concurrent program and V_C the finite set of propositional variables colled control variables which do not occur in M. By a diagram of the program M we shall understand a bipartite labeled graph $d(M,p_{en},p_{ex},P,Q)$ (for short dM) where p_{en} is a label of entry point, p_{ex} is a label of an exit point. The sets P and Q are disjoint and $P \cup Q = V_C$ is the set of all vertices of dM. The diagram dM is defined by induction with respect to the structure of M as follows:

- If M is a assignment instruction $x:=\tau$ then Fig.3.1.presents a diagram of M where $p_{en}=p_1$, $p_{ex}=p_2$, $P=\{p_1,p_2\}$ and $Q=\{q_1\}$.

- If $d(M_1,p_1^1, p_2, P^1,Q^1)$, $d(M_1,p_1^2, p_2, P^2,Q^2)$ are diagrams of M_1 and M_2 respectively (cf. Fig.3.2) such that $P^1 \cap P^2 = \{p_2\}$, $Q^1 \cap Q^2 = \emptyset$ and moreover $p_1 non\in P^1 \cup P^2$, $p_1 non\in Q^1 \cup Q^2$, then the diagram of M=**if** y **then** M_1 **else** M_2 **fi** is described in Fig.3.3 and

dM = d(**if** y **then** M_1 **else** M_2 **fi** ,p_1 ,p_2, P,Q)

where $P = P^1 \cup P^2 \cup \{p_1\}$, $Q = Q^1 \cup Q^2 \cup \{q_1\}$.

- If $d(M_1,p_1^1, p_2^1 ,P^1,Q^1)$, $d(M_2,p_1^2, p_2^2,P^2,Q^2)$ are diagrams of M_1 and M_2 respectively, cf.Fig.3.2, such that $P^1 \cap P^2 = \{p_1^2\} = \{p_2^1\}$, $Q^1 \cap Q^2 = \emptyset$ then the diagram of program $M_1;M_2$ is such that $p_{en} = p_1^1$, $p_{ex} = p_2^2, P = P^1 \cup P^2, Q = Q^1 \cup Q^2$, cf.3.4.

- If $d(M_1,p_1^1,p_2^1,P^1,Q^1)$ is a diagram and $p_2 non\in P^1$ and $p_2^1 = p_1$, $p_1^1, p_2^1 non\in P^1$, then we put

$$dM = d(\textbf{while } y \textbf{ do } M_1 \textbf{ od}, p_1,p_2 ,P,Q),$$

where $P = P^1 \cup \{p_2\}$, $Q = Q^1 \cup \{q_1\}$, $q_1 non\in Q^1$, cf. Fig.3.5.

- If $d(M_i,p_1^i, p_2^i ,P^i,Q^i)$ for $i\in n$ are diagrams of $M_1,\ldots,M_n$ respectively and $P^i \cap P^j = Q^i \cap Q^j = \emptyset$ for $i \neq j$, and $p_1,p_2 non\in \bigcup P^i$, then $d(\textbf{cobegin } M_1 \| \cdots \| M_n \textbf{ coend}, p_1,p_2,P,Q)$ is a diagram of **cobegin** $M_1 \| \ldots \| M_n$ **coend**, cf.Fig.3.6, where

$$P = \bigcup_{i\in n} P^i \cup \{p_1,p_2\}, \quad p_1,p_2 non\in \bigcup_{i\in n} P^i,$$

$$Q = \bigcup_{i\in n} Q^i \cup \{q_1,q_2\}, \quad q_1,q_2 non\in \bigcup_{i\in n} Q^i .$$

Let $dM = d(M,p_{en},p_{ex},P,Q)$ be an arbitrary diagram of a given program M. We shall call P the set of *at*-labels and the set Q the set of *in*-labels of the diagram. If q is in-labels of an action then by V_q we shall denote the set of variables occurring in this action. If p is an at-label which corresponds to some action I then by q_p we shall denote the in-label which corresponds to the same occurrence of action I.

4. SEMANTICS EXPRESSED BY MODAL FORMULAS

In this section we shall define for a given diagram dM a set of formulas $Ax^{max}(dM)$ called axioms of program M. By means of this formulas we would like to characterize MAX-semantics of program M. The set $Ax^{max}(dM)$ consists of

- local axioms Loc(dM) (which describes local behaviour of program M) and
- global axioms i.e.,

1. axiom of conflict which describes all possible conflict situations in M,
2. axiom of reachable Rea(dM), which determines the world of possible states of control,

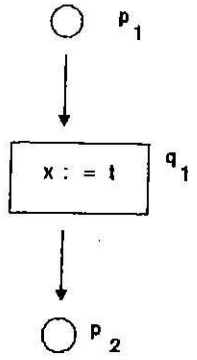


Fig. 3.1

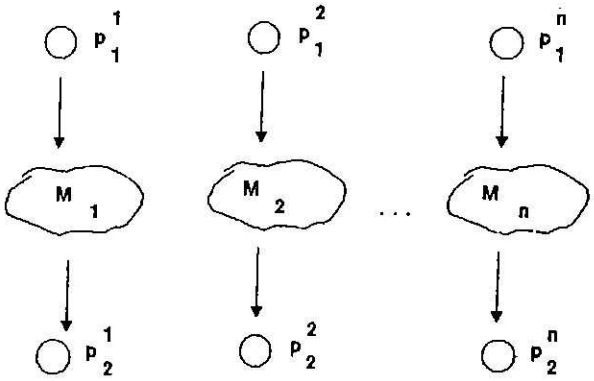


Fig. 3.2

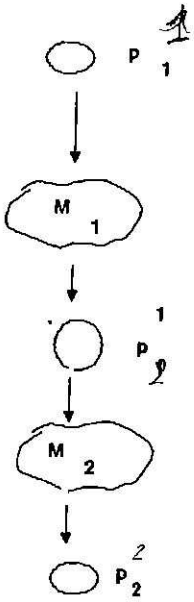


Fig.3.4

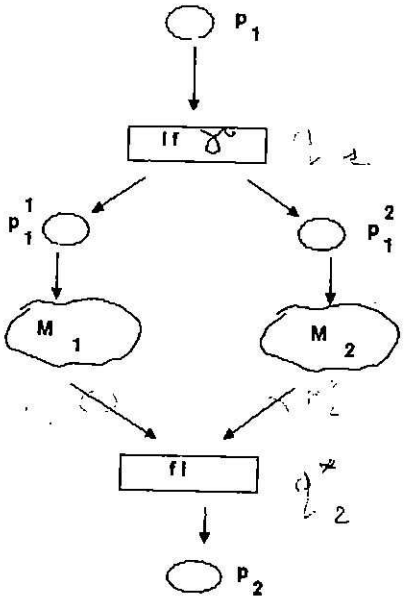


Fig.3.3

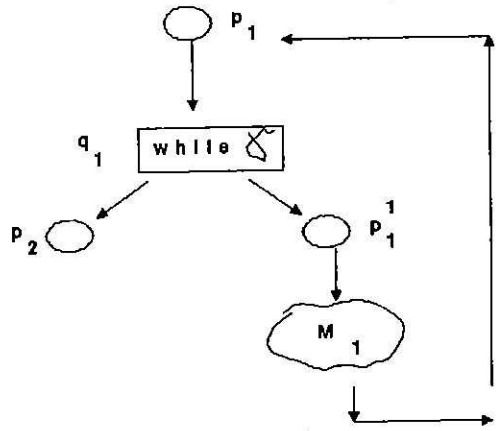

p 1
q 1
while
p 2
p 1 1
M 1

Fig. 3.5

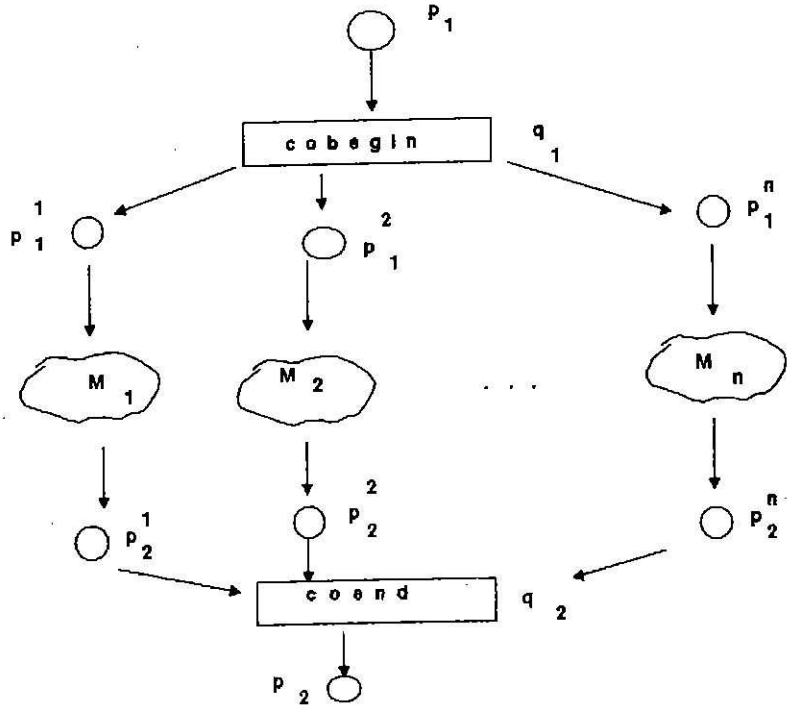

p 1
cobegin
q 1
p 1 1
p 1 2
p 1 n
M 1
M 2
M n
p 2 1
p 2 2
p 2 n
coend
q 2
p 2

Fig .3.6

3. axioms of MAX-semantics which specify the behaviour and cooperation between processes.

Local Axioms Loc(dM)

This set of formulas is defined with respect to the structure of program M.

a) Let M = $x:=\tau$ and dM be a graph described in Fig.3.1. Then Loc(dM) consists of the following schemas of formulas

(4.1) $p_1 \Rightarrow \Box(p_1 \vee \neg p_1 \wedge q_1)$

(4.2) $(q_1 \vee \alpha(x/\tau)) \Rightarrow \Box(q_1 \wedge \alpha(x/\tau) \vee \neg q_1 \wedge p_2 \wedge \alpha(x))$

where x is the only free variable in α .

b) Let M = **if** γ **then** M_1 **else** M_2 **fi** and let dM is described in Fig.3.3. Then Loc(dM) consists of Loc(dM_1), Loc(dM_2) and the following schemas of formulas

(4.3) $p_1 \Rightarrow \Box(p_1 \vee q_1 \wedge \neg p_1)$

(4.4) $(q_1 \wedge \gamma \wedge \alpha) \Rightarrow \Box((q_1 \wedge \gamma \vee \neg q_1 \wedge p_1^1) \wedge \alpha)$

(4.5) $(q_1 \wedge \neg\gamma \wedge \alpha) \Rightarrow \Box((q_1 \wedge \neg\gamma \vee \neg q_1 \wedge p_1^2) \wedge \alpha)$

for arbitrary formula α such that $V(\alpha) \subset V(\gamma)$

c) Let M = $M_1;M_2$ and let dM be described in Fig.3.4. Then Loc(dM) consists of Loc(dM_1), Loc(dM_2).

d) Let M = **while** γ **do** M_1 **od** and let dM be described in Fig.3.5. Then Loc(dM) consists of Loc(dM_1) and the following schemas of formulas

(4.6) $p_1 \Rightarrow \Box(p_1 \vee q_1 \wedge \neg p_1)$

(4.7) $(q_1 \wedge \gamma \wedge \alpha) \Rightarrow \Box((q_1 \wedge \gamma \vee \neg q_1 \wedge p_1^1) \wedge \alpha)$

(4.8) $(q_1 \wedge \neg\gamma \wedge \alpha) \Rightarrow \Box((q_1 \wedge \neg\gamma \vee \neg q_1 \wedge p_2) \wedge \alpha)$

for arbitrary formula α such that $V(\alpha) \subset V(\gamma)$

e) Let M = **cobegin** $M_1 \| \cdots \| M_n$ **coend** and let dM be described in Fig.3.6. Then Loc(dM) consists of $\bigcup_{i \le n}$ Loc(dM_i) and the following schemas

(4.9) $p_1 \Rightarrow \Box(q_1 \wedge \neg p_1)$

(4.10) $q_1 \Rightarrow \Box(\neg q_1 \wedge p_1^1 \wedge p_1^2 \wedge \ldots \wedge p_1^n)$

(4.11) $(p_1^1 \wedge p_1^2 \wedge \ldots \wedge p_1^n) \Rightarrow \Box(q_2 \wedge \neg p_1^1 \wedge \ldots \wedge \neg p_1^n)$

(4.12) $q_2 \Rightarrow \Box(\neg q_2 \wedge p_2)$

Axioms of conflicts

Let us denote by Conf(dM) the alternative of all formulas of the form $(q \wedge q')$ such that ~~and~~ q is in-label of $x:=\tau$, for some τ and q' is an in-label

either of "if γ" and $x \in V(\gamma)$
or of "while γ" and $x \in V(\gamma)$
or of "$y:=\eta$" and $x \in V(\eta)$
or of "$x:=\eta$".

Axiom of conflict is then in the following form

(4.13) $\neg$ Conf(dM)

This formula indicates actions which cannot be executed simultaneously.

Axioms of MAX-semantics

For the sake of simplicity let us assume the following denotations

$$at(A) =^{df} \bigwedge_{p \in A} p \wedge \bigwedge_{p \in P-A} \neg p \quad \text{and} \quad in(I) =^{df} \bigwedge_{q \in I} q \wedge \bigwedge_{q \in Q-I} \neg q$$

where A and I are arbitrary subsets of P and Q respectively.

(4.14) $at(A) \Rightarrow (max \equiv \bigwedge_{p \in A} conf(q_p/P))$

(4.15) $(at(A) \wedge in(I) \wedge max \wedge \alpha) \Rightarrow \Box(\neg in(I) \wedge \alpha)$
for arbitrary α such that $V(\alpha) \subset A \cup (Q-I)$

(4.16) $(at(A) \wedge in(I) \wedge max \wedge \alpha) \Rightarrow (\Diamond in(I-J) \wedge \neg\Diamond(in(I-J) \wedge \neg\alpha))$
for arbitrary α such that $V(\alpha) \subset V - \bigcup\{V_q : q \in J\}$ and $J \subset I$

(4.17) $(at(A) \wedge in(I) \wedge \neg max \wedge \alpha) \Rightarrow \Box(max \wedge \neg at(A) \wedge \alpha)$
for arbitrary α such that $V(\alpha) \subset V \cup (P-A) \cup I$

(4.18) $(at(A) \wedge in(I) \wedge \neg max) \Rightarrow \Diamond at(A-J)$ for arbitrary $J \subset A$.

The first group of axioms of MAX-semantics (4.14) express what we mean by maximality. Propositional variable *max* indicates that the set of in-labels cannot be enriched by any of at-labels without conflict . The set of instructions under execution is maximal if and only if each possible extension leads to conflict.

The second group of axioms (4.15) expresses that being in the state of maximal engagement of processors at least one active instruction must be executed to obtain the next state.

The last group (4.17) states that from a state of not maxial engagement of processors we necessarily reach a state of maximal

engagement of processors. Moreover by (4.18) no individual variable is changed in such situation. The possible changes will take place only inside the set A of at-labels ,cf. (4.17).

Axioms of reachability

This axiom denoted by Rea(dM) is defined by induction with respect to the structure of program M. For the sake of simplicity we shall use a logical operator $|$ with the following meaning

$$\alpha_1 |\dots|\alpha_n \equiv \bigvee_{1\le i\le n} \Big(\alpha_i \wedge \bigwedge_{j\neq i} \neg\alpha_j \Big)$$

(e.g. $p|q|r \equiv (p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r)$).

Rea(dM) $=^{df} p_1|q_1|p_2$ for dM presented in Fig. 3.1

Rea(dM) $=^{df} p_1|q_1|\text{Rea}(dM_1)|\text{Rea}(dM_2)|p_2$ for dM presented in Fig. 3.3

If Rea(dM_1)= $\alpha|p_2^1$ and Rea(dM_2)= $p_2^1|\beta$ then

Rea(dM) $=^{df} \alpha|p_2^1|\beta$ for dM presented in Fig. 3.4

Rea(dM) $=^{df} p_1|q_1|\text{Rea}(dM_1)|p_2$ for dM presented in Fig. 3.5

Rea(dM) $=^{df} p_1|q_1|\text{Rea}(dM_1)|\dots|\text{Rea}(dM_n)|q_2|p_2$ for dM presented in Fig. 3.6

The role of this axiom is to describe the possible states of control in all executions of program.

5. FUNDAMENTAL MODEL

Let us consider an arbitrary fixed data structure **A** and the set **C** of all configurations which occur in any computation of a fixed program M in **A**. On the base of this set and the relation $\max_{\to}$, defined as in section 2, we shall construct below a semantic structure

$$\mathbf{Comp(A)} = \langle S, R, w \rangle$$

which we hope to prove is a model of the set of corresponding modal formulas presented in the previous section.

Let dM = $d(M,P_{en},P_{ex},P,Q)$ be a fixed diagram of the program M. For arbitrary configuration $c=\langle v,K\rangle$, $c\in C$, such that IN is the set of all instructions under execution in K and AT is the set of all instructions ready for execution in K, we put

$v^+(z) = v(z)$ for arbitrary individual variable $z\in V$,
$v^+(q) = 1$ iff q is an in-label of an instruction from IN,
$v^+(p) = 1$ iff p is an at label of an instruction from AT,
$v^+(max) = 1$ iff the set IN is a maximal nonconflict set.

Let *trans* be a mapping defined for arbitrary configuration $c=\langle v,K\rangle\in C$ by the equality

$$trans(\langle v,K\rangle) = \langle v^+,K\rangle.$$

We put

$S = trans(C)$
$R = \{ (trans(c_1),trans(c_2)): c_1 \overset{max}{\to} c_2 , c_1,c_2\in C \}$
$w(s) = v^+$ for arbitrary state $s=\langle v^+,K\rangle$, $s\in S$.

REMARK It is easy to see that *trans* is a one-to-one mapping $trans: C \to S$ and moreover it is an isomorphism which maps the structure $\langle C,\overset{max}{\to}\rangle$ onto $\langle S,R\rangle$.

THEOREM 5.1
For arbitrary data structure **A** , **Comp(A)** is a model of the set Ax(dM), i.e. **Comp(A)** $\models$ Ax(dM).

PROOF
Let us consider an arbitrary fixed state s in **Comp(A)**. By the definition there is exactly one configuration c such that $trans(c) = s$. Let us assume that

(5.1) $$s \models (at(A) \wedge in(I))$$

for some sets $A\subset P$, $I\subset Q$.
Hence each instruction which at-label belongs to A is marked by $\sharp$ and each instruction which in-label belongs to I is marked by $\diamond$. Let us denote the set of all instructions marked by $\diamond$ and $\sharp$ by IN and AT respectively. Below we would like to prove that all axioms Ax(dM) are valid in s.
a) Suppose s *non*$\models\neg$Conf. Thus $s \models$ Conf, i.e., there are at least two in-labels q_1 and q_2 which corresponds to K_1 and K_2 in the program K such that $s \models (q_1 \wedge q_2)$. Hence $q_1,q_2\in A$ and $K_1,K_2 \in$IN. By the construction of the formula Conf, the instructions K_1,K_2 are

in conflict. The set In is the conflict set too and therefore $\langle v,K\rangle$ does not belong to C, a contradiction. This ends the proof of validity of axiom of conflict.

b) By (5.1) the predecessor of the formula (4.14) is valid. If for some $p \in A$, s $non\models$ $Conf(p/q_p)$ then the set IN extended by action with the label p, create a nonconflict set. Since $q_p non \in I$, the set In is not a maximal set, i.e., s $non\models$ max.

Conversely, if for arbitrary p,

$$s \models Conf(p/q_p)$$

then the instruction labeled by p is in conflict with each instruction marked by $\diamond$. Hence the set IN cannot be extended and, in view of previous considerations, IN is a maximal nonconflict set which implies $s \models max$. This ends the proof of validity of formula (4.14).

c) Assume now that $s \models max$. Let $c' = \langle v',K'\rangle$ be a successor of $\langle v,K\rangle$ and $s' = trans(c')$. By the definition of semantics, cf. case (2.1), the instructions marked by star $\diamond$ in K are still marked by $\diamond$ in K', i.e., $s' \models p$ for $p \in A$.

Moreover, K' is obtained from K by removing executed in this step actions. The in-labels of this instructions will not be valid and therefore $s' \models \neg\, in(I)$. This proves validity of (4.16) and (4.15).

d) Assume that s $non\models$ max. Hence IN is not a maximal set. By the definition of semantics, cf. (2.2), it is necessary to extend IN to a maximal noncoflict set. Hence in all successors of s, all instructions marked by $\diamond$ will be still marked by $\diamond$ and some instructions marked by $\sharp$ will change mark. Thus in each immediate successor s' of s we have

$$s' \models (\neg\, at(A) \wedge in(I) \wedge max)$$

Furthermore, since valuation of individual variables in s' is identical with v,

$$s \models {}^{\circ}\alpha \Rightarrow s' \models \alpha$$

for arbitrary formula α with individual variables only, i.e., the formulas (4.17) and (4.18) are valid.

e) The validity of axiom of reachable follows from the definition of semantics and the definition of the formula Rea. The base idea expressed in this formula is that occurrence of an action cannot be

simultaneously marked by ◦ and ♯. Moreover, in each sequential program at least one action is marked.
The exact proof, by induction with respect to the structure of program M, is omitted.

f) Let us consider finally the local axioms. Suppose

(5.2) $$s \models p_1$$

and p_1 occurs in the diagram dM in the context presented in Fig.3.1. By (5.2) the instruction $x:=\tau$ is marked by ♯ and therefore occurs in AT. If $s \models max$ then by the above considerations and (5.2) for all successors s', $s' \models p_1$.
If s $non \models max$ then possible that $(x:=\tau)$ is chosen for execution. Thus in all possible successors s', $s' \models (p_1 \vee q_1 \wedge \neg p_1)$.
This proves validity of the formula (4.1).
Suppose

(5.3) $$s \models q_1$$

and let $\alpha(x)$ be a formula with one free individual variable x, such that $s \models \alpha(x/\tau)$.

If s $non \models max$ then from the above considerations (cf. part d of the proof) $s' \models (q_1 \wedge \alpha(x/\tau))$ for arbitrary successor s' of s. If $s \models max$ then the instruction $(x:=\tau)$ is possibly executed in this step of computation. Hence its in-label disappears from the set I and the successor of $(x:=\tau)$ will obtain mark ♯. Moreover, since $v(x/\tau) \models \alpha(x)$, we have for all such successors s' of s, $s' \models \alpha(x)$. Thus finally $s' \models (\neg q_1 \wedge p_2 \wedge \alpha(x))$.

For $s \models max$ it is also possible that $(x:=\tau)$ is not executed in this step of computation. Hence in such successors s' of s, $s' \models q_1$. The executed in this step instructions cannot change any variable from $V(\tau) \cup \{x\}$, since of conflict. Thus $s' \models (q_1 \wedge \alpha(x/\tau))$. Both cases together proves the formula (4.2). This ends the proof of the theorem 5.1. □

As a simple consequence of the above proof we obtain the following observation (cf. Appendix).

LEMMA 5.2

For arbitrary data structure A and an arbitrary initial configuration $\langle v,*M\rangle$ in A, the substructure **Comp**(A,v) determined by $\langle v,*M\rangle$ is a model of Ax(dM). □

DEFINITION

We shall call **Comp**(A,v) a MAX-computational model determined by A and v. □

Similarly we can construct a computational model based on successorship relation $\text{arb}_{\to}$. The resulting structure denoted by $\mathbf{Comp}^{\text{arb}}(v,A)$, let us call ARB-computational model determined by A and v.

THEOREM 5.3

For arbitrary data structure A and an arbitrary valuation v in A, $\mathbf{Comp}^{\text{arb}}(v,A)$ is a model of $\text{Ax}^{\text{arb}}(\text{dM})$. □

6. AXIOMS DETERMINES SEMANTICS

In this section we try to argue that the converse to the theorem 5.1 is in some sence also valid. Namely we would like to prove that an arbitrary model of the set of axioms defined in section 2 can be restricted to a computational model.

Let **M**(A) $=\langle S,R,w\rangle$ be an arbitrary homogeneous model for $\text{Ax}^{\max}(\text{dM})$ such that *non* **M**(A) $\models \neg p_{\text{en}}$ and $d(M)=\langle M,p_{\text{en}},p_{\text{ex}},P,Q\rangle$. We assume moreover that w is a one-to-one mapping. We shall call such model *proper for the diagram* dM.

For $s_0 \in S$ such that $s_0 \models p_{\text{en}}$, let $\mathbf{M}_0(A,s_0)=\langle S_0,R_0,w_0\rangle$ be a submodel of **M**(A) determined by s_0.

THEOREM 6.1

Let $v_0=w(s_0)$ and **Comp**(A,v_0) be a computational model of $\text{Ax}^{\max}(\text{dM})$. Then **M**(A,$s_0$) is isomorphic to **Comp**(A,v_0).

PROOF

Let us put $h(s) =^{\text{df}} w(s)$ for all $s \in S$.

Thus by definition, h is one-to-one mapping. We would like to prove

(6.1) $(\forall s \in S_0)\quad w(s) \in \mathbf{Comp}(\mathbf{A}, v_0)$

(6.2) $(\forall v \in \mathbf{Comp}(\mathbf{A}, v_0))\ (\exists s \in S_0)\quad w(s) = v.$

To prove (6.1) we shall show that for arbitrary state s, if $w_0(s) \in \mathbf{Comp}(\mathbf{A}, v_0)$ then for all $s' \in R_0(s)$, $w_0(s') \in \mathbf{Comp}(\mathbf{A}, v_0)$.
Let s be an arbitrary state of $\mathbf{M}(\mathbf{A}, s_0)$ such that $w_0(s) = v$ and $w_0(s) \in \mathbf{Comp}(\mathbf{A}, v_0)$.

Case 1

$s \models (at(A) \wedge in(I) \wedge max)$ for some fixed ACP and ICQ.

Let s' be a fixed successor of s, sR_0s'. By (4.15), $w(s')$ and v are equal on the set $Au(Q-I)$. Moreover it is necessary that some variables from the set I have been changed while moving from s to s'. Assume that $s' \models in(I-J)$ for some $J \subset I$, $nonJ = \emptyset$.

If $q \in J$ and q is an in-label of the instruction $(x := \tau)$ (cf. Fig.3.1) then, by local axioms, for arbitrary formula α such that $V(\alpha) = \{x\}$,

$s \models \alpha(x/\tau)$ implies $s' \models p_2 \wedge \alpha(x)$.

Hence $w_0(s')(x) = \tau_A(v)$ and $w_0(s') \models p_2$. By axioms of reachability $w_0(s') \models \neg p_1$.
Similarly we can analyze other in-labels $q \in J$.

By axioms (4.16) no other individual variable from $V - \cup\{V_q : q \in J\}$ will change value and by local axioms all changes are connected with assignment instructions labeled by $q \in J$. Finally, by (4.14) the set of instructions indicated by I is a maximal nonconflict set. It may be observed now easily that $w_0(s')$ is a valuation obtained from v according to the rule (2.1) where the set of executed instructions are marked by in-labels from J. Hence $v^{\max} \to w_0(s')$ and $w_0(s') \in \mathbf{Comp}(\mathbf{A}, v_0)$.

Case 2

$s \models (at(A) \wedge in(I) \wedge \neg max)$ for some ACP and ICQ.

Let s' be a fixed successor of s, sR_0s'. By (4.17), $w(s')$ and v are equal on the set $V \cup I \cup (P - A)$. Thus while moving from s to s' no individual variable will change, all instructions under execution in v and all instructions not ready for execution in v will remain in $w_0(s')$. Furthermore, by (4.17) all the possible changes are connected with the set A.

Let $s' \models at(A\text{-}J)$. Analysing local axioms we come to conclusion that $w_0(s') \models q$ for all q being in-label corresponding to at-label $p \in J$. By (4.14) the set of instructions marked by in-labels from I is not maximal set and by (4.17) the set $\{q \in Q: w_0(s') \models q\}$ is maximal nonconflict set. This proves that the valuation $w_0(s')$ is obtained from v according to the rule (2.2) of the definition of MAX-semantics.

To prove (6.2) it is enough to show that for arbitrary valuation $v \in \mathbf{Comp}(\mathbf{A},v_0)$, if $w_0(s)=v$ for some s, then for each v' such that $v \overset{max}{\rightarrow} v'$ there exists a state s' in S_0 with $w_0(s')=v$. The proof similar to the presented above is omitted.

To end the proof of the theorem 6.1 let us remark that due to the construction

$$s \; R_0 \; s' \qquad \text{iff} \quad h(s) \overset{max}{\rightarrow} h(s'),$$

Thus h is an isomorphism which transforms $\mathbf{M}(\mathbf{A},s_0)$ onto $\mathbf{Comp}(\mathbf{A},v_0)$. □

Analogous result can be proved for the other concept of concurrency.

THEOREM 6.2
For arbitrary data structure **A**, if $\mathbf{M}(\mathbf{A},s)$ is an arbitrary proper model of $Ax^{arb}(dM)$ determined by s then $\mathbf{M}(\mathbf{A},s)$ is isomorphic to $\mathbf{Comp}^{arb}(\mathbf{A},v(s))$. □

7. APPENDIX

We are going to present here a modal logic we are dealing with in this paper.

Let L^m denote an extension of the language L by a finite set of propositional variables V_C and modal operators necessary □ and possible ◇. Hence the set F of all formulas of L^m contains modal formulas of the form □α and ◇α for arbitrary α from L^m, apart of classical open formulas. The sets of terms of L and L^m are identical, we shall denote it by T.

Let **A** be a data structure for L, *i.e.*, a relational system of the same signature as the language L. We shall assume that the reader is familiar with the semantics of the first order language. Let us mention here only that $\tau_A(v)$ denotes a value of a term τ in the structure **A** at the valuation v and $\mathbf{A},v \models \alpha$ denotes that v satisfies α in **A**.

DEFINITION
By the semantic structure of the language L^m we shall understand a Kripke-like structure $\mathbf{H} = \langle S,\{\mathbf{A}(s):s\in S\},R,w\rangle$ such that S is a nonempty set of states, R is a binary relation in S, $R \subset S \times S$, $\mathbf{A}(s)$ is for every $s\in S$ a data structure for L^m and w is a function defined on S which assigns valuation in $\mathbf{A}(s)$ and Boolean algebra to arbitrary state s .
The meaning of expressions of L^m is then defined as follows:

$$\tau_M(s) = \tau_{\mathbf{A}(s)}(w(s))$$

$$\mathbf{H},s \models \alpha \equiv \mathbf{A}(s),w(s) \models \alpha$$

for arbitrary term τ and formula α from L. Moreover

$$\mathbf{H},s \models \Box\alpha \equiv R(s) \neq \emptyset \text{ and } (\forall s')(s'\in R(s) \Rightarrow \mathbf{H},s' \models \alpha)$$

$$\mathbf{H},s \models \Diamond\alpha \equiv (\exists s')(s'\in R(s) \wedge \mathbf{H},s' \models \alpha)$$

for arbitrary $\alpha\in L^m$. □

Let us consider a formal system **L** determined by the set Ax of axioms and three rules of inference. The set Ax contains all axioms of classical propositional calculus and the following schemas

$$(\Box\alpha \Rightarrow \Diamond\alpha)$$

$$\Box(\alpha \wedge \beta) \equiv (\Box\alpha \wedge \Box\beta)$$

$$\Diamond(\alpha \wedge \beta) \Rightarrow (\Diamond\alpha \wedge \Diamond\beta)$$

$$(\Box\alpha \vee \Box\alpha) \Rightarrow \Box(\alpha \vee \beta)$$

$$\Diamond(\alpha \vee \beta) \equiv (\Diamond\alpha \vee \Diamond\beta)$$

$$\neg\Box\alpha \equiv (\neg\Diamond\mathbf{true} \wedge \Diamond\neg\alpha)$$

$$\neg\Diamond\alpha \equiv (\neg\Box\mathbf{true} \vee \Box\neg\alpha)$$

$$\neg\Box\mathbf{false}$$

wynika tautologicznie

The set of rules contains modus ponens and

$$\frac{(\alpha \Rightarrow \beta)}{(\Diamond\alpha \Rightarrow \Diamond\beta)} \qquad \frac{(\alpha \Rightarrow \beta)}{(\Box\alpha \Rightarrow \Box\beta)}$$

For arbitrary set of formulas Z , $Z \models \alpha$ denotes that α is a semantic consequence of the set of formulas Z, and $Z \vdash \alpha$ denotes that α is provable from Z, i.e. is a syntactic consequence of Z.

Let Z be a consistent set of formulas and let α_0 be a formula of the language L^m such that

(7.1) $$non\ Z \vdash \alpha_0 .$$

The Lindenbaum algebra $\langle F/\approx, \wedge, \vee, - \rangle$ of the theory determined by Z is therefore nondegenerate Boolean algebra, where

$\alpha \approx \beta$ iff $Z \vdash (\alpha \Rightarrow \beta) \wedge (\alpha \Rightarrow \beta)$

and for arbitrary $[\alpha] =^{df} \{\gamma : \alpha \approx \gamma\}$ and $[\beta] =^{df} \{\gamma : \gamma \approx \beta\}$

$$[\alpha] \wedge [\beta] = [\alpha] \wedge [\beta]$$
$$[\alpha] \vee [\beta] = [\alpha] \vee [\beta]$$
$$-[\alpha] = [\neg\alpha]$$

Moreover for arbitrary formula α,

(7.2) $non\ Z \vdash \alpha$ iff $[\neg\alpha] \neq [\alpha] = 0$.

By Kuratowski-Zorn lemma for every non-zero element β there is a maximal filter in the Lindenbaum algebra which contains β. Hence by (7.1) and (7.2) the set S of all maximal filters is not empty.

For a given filter F, let $A(F)$ be a data structure for L in the set of all terms T such that for arbitrary functor φ and arbitrary predicate ρ in L

$$\varphi_{A(F)}(\tau_1,\ldots,\tau_n) = \varphi(\tau_1,\ldots,\tau_n)$$
$$\rho_{A(F)}(\tau_1,\ldots,\tau_n) \quad \text{iff} \quad [\rho(\tau_1,\ldots,\tau_n)] \in F.$$

Let $M = \langle S, \{A(s): s \in S\}, R, \Vdash \rangle$ be a semantic structure such that S is the set of all maximal filters in the Lindenbaum algebra described above,

$R = \{(s_1,s_2)$: $[\Box true] \in s_1$ and $(\forall\alpha)$ $([\Box\alpha] \in s_1 \Rightarrow [\alpha] \in s_2)$ $\}$,

$W(s)$ is a valuation v such that $v(x)=x$ for individual variable $x\in V$ and $v(p)=1$ iff $[p]\in s$, $s\in S$.

THEOREM 7.1

For every formula α and for every $s\in S$,

$$\mathbf{M},s \models \alpha \text{ iff } [\alpha]\in s.$$

PROOF

The proof is by induction with respect to the structure of α. When α is a propositional variable or elementary formula, the proof follows from the definition of semantic structure. When α is of the form $(\gamma \vee \beta)$, $(\gamma \wedge \beta)$ or $\neg\gamma$ the proof follows easily from properties of maximal filters.

Let us consider more strictly the formula $\Diamond\beta$ assuming that the theorem was proved already for β.

Suppose $\mathbf{M},s \models \Diamond\beta$ for some fixed state s. By the definition of semantics, there is at least one state s' in S such that

(7.3) $$sRs' \text{ and } \mathbf{M},s' \models \beta.$$

If $[\Diamond\beta]non\in s$ then $[\neg\Diamond\beta]\in s$, since s is a maximal filter. Hence either $[\neg\Box\text{true}]\in s$ or $[\Box\neg\beta]\in s$. The first case contradicts (7.3). In the second case we have by the definition $[\beta]non\in s'$ and by inductive assumption $\mathbf{M},s' non\models \beta$ which contradicts the second part of (7.3).

Conversely, suppose that

(7.4) $$[\Diamond\beta]\in s.$$

We shall tray to determine a state s' such that $[\beta]\in s'$. To this aim, let us consider the set $F = \{[\gamma]: [\Box\gamma]\in s\}$.

Clearly F is a filter. Suppose

(7.5) $$[\alpha_1],\ldots,[\alpha_n]\in F \text{ and } [\alpha_1]\wedge\ldots\wedge[\alpha_n] \wedge [\beta] = 0.$$

Thus by property (7.2) of Lindenbaum algebra

$$Z \vdash ((\alpha_1 \wedge\ldots\wedge \alpha_n) \Rightarrow \neg\beta).$$

Hence applying one of inference rules we obtain

$$Z \vdash \Box(\alpha_1 \wedge\ldots\wedge \alpha_n) \Rightarrow \Box\neg\beta$$

and by (7.5) $[\Box\neg\beta]\in s$ (i.e. $[\neg\Diamond\beta]\in s$) contrary to (7.4). This proves that the considered set F has finite intersection property. Thus the set $F\cup\{\beta\}$ can be extended to a maximal filter. Let us denote it by s'. By the definition and assumption (7.4) $(s,s')\in R$ and

$[\beta] \in s'$. Hence by the inductive hypothesis $\mathbf{M},s' \models \beta$ and as a consequence $\mathbf{M},s \models \Diamond\beta$.

The analogous considerations for formula $\Box\beta$ are omitted. □

The above theorem can be reformulated to as every consistent set of formulas in L has a model. This allow to proves the following completeness theorem.

THEOREM 7.2
For arbitrary set of formulas Z and for an arbitrary formula α the following holds

$Z \models \alpha$ if and only if $Z \vdash \alpha$.

We end this auxiliary section with the following observation. Let $\mathbf{M} = \langle S,\{\mathbf{A}(s):s\in S\},R,w\rangle$ be an arbitrary semantical structure of the language L^m. For a given state s_0 of S let $\mathbf{M}_0(s_0)$ be a substructure $\langle S_0,\{\mathbf{A}(s):s\in S_0\},R_0,w_0\rangle$ defined as follows

$S_0 = \{s\in S:\ s_0\ R^{rtc}\ s\ \}$,

where R^{rtc} is the reflexive and transitive closure of R ,

$R_0 = R\ /S_0 X S_0$ and $w_0 = w_\phi/S_0$.

LEMMA 7.3
For arbitrary set of formulas Z, if $\mathbf{M}$ is a model of Z then the substructure $\mathbf{M}_0(s_0)$ is also a model of Z. □

The class of models of the language L^m is very reach. To our purposes it is convenient to restrict this class to models concerning one data structure only.

DEFINITION
The semantic structure $\mathbf{M} = \langle S,\{\mathbf{A}(s):\ s\in S\},R,w\rangle$ we shall call homogeneous if all data structures $\mathbf{A}(s)$ are identical with $\mathbf{A}$ for every $s\in S$. We shall denote such structure by $\mathbf{M}(\mathbf{A})$. □

REFERENCES

[AFR] Apt K.R., Francez N., de Roever W.P., A Proof System for Communicating Sequential Processes, ACM TOPLAS, vol.2,No.3 (1980), 361-385

[BD] Best E., Devillers R., Concurrent Behaviour: Sequences, Processes and Programming Languages, GMD-Studien No.99 (1985)

[EM] Enjalbert P., Michel M., Many-sorted Temporal Logic for Multiprocesses Systems, Proc. MFCS'84 (Chytil M. and Koubek V. eds) in LNCS 176 ,1984, 273-282

[F] Floyd R.W., Assigning Meanings to Programs, Proc.AMS (1967)

[H] Hoare C.A.R., Communicating Sequential Processes, CACM 21 (1978), 666-677

[K] Kripke S., Semantical analysis of modal logic II. The theory of Models, Amsterdam North-Holland 1965

[L] Lamport L., Schneider F., The "Hoare Logic" of CSP and All That, TOPLAS 6 (1984), 281-295

[MS] Mirkowska G., Salwicki A., Algorithmic Logic, Warsaw, PWN -Reidel Publ. (1987)

[OG] Owicki S., Gries D., An Axiomatic Proof Technique for Parallel Programs, Acta Informatica 6 (1976), 319-340

[SM] Salwicki A., Muldner T., On Algorithmic Properties of Concurrent Programs , in Logics of programs, Zurich 1979 (E.Engeler ed.) LNCS 125 (1981) Springer-Verlag, 169-197

[R] de Roever W.P., The quest for compositionality - a survey of assertionbased Proof Systems for Concurrent Programs, Part I: Concurrency Based on Shared Variables, Proc. of the IFIP Working Conf. 1985 "The Role of Abstract Models in Computer Science" (E. J.Neuhold ed.) North-Holland