PRACE IPI PAN • ICS PAS REPORTS

Józef Winkowski

A distributed implementation of Petri nets

518

June 1983

WARSZAWA

INSTYTUT PODSTAW INFORMATYKI POLSKIEJ AKADEMII NAUK
INSTITUTE OF COMPUTER SCIENCE POLISH ACADEMY OF SCIENCES
00-901 WARSAW, P.O. Box 22, POLAND

Józef Winkowski

A DISTRIBUTED IMPLEMENTATION OF PETRI NETS

518

Warsaw, June 1983

Pracę zgłosił Andrzej Blikle

Mailing address: Józef Winkowski
Institute of Computer Science
Polish Academy of Sciences
P.O. Box 22
00-901 Warszawa PKiN

ISSN 0138-0648

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 320 egz. Ark wyd. 2,00; ark. druk. 3,25.
Papier offset. kl. III, 70 g, 70 x 100. Oddano do druku w czerwcu 1983 r. W. D. N. Zam. nr 421/83

Abstract . Содержание . Streszczenie

The paper is devoted to implementation of such systems of activities which can be represented by Petri nets. It is shown that systems of this type can be executed by distributed networks of interconnected modules controlling places and transitions of the corresponding nets. The modules executing a net and the links between them depend only on local properties of the net, not on global ones. Due to this feature the speed at which each particular activity is executed does not depend on the size of entire system of activities.

О распределенной реализации сетей Пэтри

Работа касается реализации систем действий описываемых сетями Пэтри. Показывается, что такие системы реализуемы распределенными сетями модулей контролирующих места и переходы соответствующих сетей Пэтри. Модули реализующие данную сеть Пэтри и связи между ними зависят только от локальных свойств этой сети. Благодаря этому скорости выполнения действий не зависят от размера системы.

O implementacji sieci Petri przez systemy rozproszone

Praca jest poświęcona implementacji systemów czynności opisywalnych sieciami Petri. Pokazano, że takie systemy mogą być realizowane przez rozproszone sieci połączonych modułów kontrolujących miejsca i przejścia odpowiednich sieci. Moduły wykonujące sieci i ich połączenia zależą jedynie od lokalnych własności sieci, a nie od własności globalnych. Dzięki temu szybkości, z jakimi są wykonywane poszczególne czynności, nie zależą od rozmiarów systemu.

1. Introduction

The purpose of this paper is to show that a system of activities can be implemented such that:

- each activity is executed (possibly repeatedly) by a module which depends only on this activity, not on the entire system,
- the transfer of information from one activity to another is realized by a communication between the corresponding modules,
- each module reacts only on its own state and on the state of its communication links.

The reason of tending to such implementation is twofold:

- each activity of the system can be executed at a speed which does not depend on the system size,
- the implementation is fault-resistant in the sense that a fault in one module does not necessarily lead to a fall of entire system.

The requirements imposed on implementation have the following consequences:

- any central co-ordination or synchronization of modules and any global data are not allowed,
- the modules can exchange information only with the aid of established communication lines.

In other words, such requirements can be fulfilled only by a distributed implementation.

In our considerations we assume that systems of activities are represented by Petri nets and concentrate on such nets.

We recall that a Petri net is a bipartite directed graph, i.e., a directed graph P whose set of nodes is partitioned into two subsets: a subset B of circles, called places, and a subset E of boxes, called transitions, each arc connecting only nodes of different types. Formally, $P=(B,E,F)$, where $F\subseteq B\times E\cup E\times B$ (cf. [2]). We assume that P is finite, i.e., B and E are finite.

If an arc is directed from node x to node y (either from a place to a transition or a transition to a place) then we write xFy, x is called an input to y, and y is called an output of x. Input x to y (resp.: output y of x) is said to be pure if it is not an output of y (resp.: input to x). By Fx (resp.: by xF) we denote the set of all inputs to x (resp.: of all outputs of x). Two transitions e and e′ with $(Fe\cup eF)\cap(Fe'\cup e'F)=\emptyset$, i.e. with disjoint sets of adjacent places, are said to be independent.

Each place may carry a number of markers, called tokens. This gives a distribution of tokens, called a marking. Formally, a marking is a mapping $m:B\longrightarrow\{0,1,\dots\}$, where $m(b)$ represents the number of tokens in place b.

Sometimes it is convenient to assume that the number of tokens in a place b cannot exceed a certain limit, called the capacity of b, and denoted capacity(b). This leads one to the concept of a *restricted* Petri net, where only those markings m are admitted which satisfy m(b)$\leqslant$ *capacity*(b) for all places b. Formally, such net is P=(B,E,F,*capacity*), where B,E,F are as before and, in addition, we have a mapping *capacity*:B$\longrightarrow$$\{0,1,\ldots,+\infty\}$.

The usual (unrestricted) Petri nets can be regarded as restricted ones with *capacity*(b)=+∞ for all places b. The so called *condition/event nets* are restricted Petri nets with *capacity*(b)= for all places b (the name comes from the fact that places of such nets represent *conditions* which may hold or not, and transitions represent *events* which may occur and change the holding of conditions).

That a net P is a representation of a system of activities is reflected by the concept of execution.

The execution of P starts with a marking and changes it according to a precise principle, called the *firing rule*. This principle is as follows.

A transition e is said to be *fireable* (or *enabled*) under a marking m if each input place b$\in$Fe carries at least one token, and the number of tokens in each pure output place b$\in$eF-Fe is less than *capacity*(b).

Being fireable under a marking m, a transition e may *fire* (or be *executed*, or it may *occur*). The firing is assumed to change m to another marking, denoted me, where:

$$me(b) = \begin{cases} m(b)-1 & \text{for } b \in Fe-eF \\ m(b)+1 & \text{for } b \in eF-Fe \\ m(b) & \text{for other places .} \end{cases}$$

Such firing is regarded to be an <u>indivisible</u> operation in the sense that it cannot be disturbed by firing any other transition which is not independent of e, i.e., with adjacent places in $Fe \cup eF$. As a consequence, if several transitions are enabled that are not independent then a <u>conflict</u> arises which must be solved in order to decide which transitions may fire. The decisions of this type are assumed to be indeterministic.

Each finite initial segment of execution of P starting from a marking m is represented by a string $x=e_1 \dots e_k$, called a <u>firing sequence</u>, such that: e_1 is fireable under m, e_2 is fireable under $me_1, \dots,$ and e_k is fireable under $me_1 \dots e_{k-1}$. The order in such string does not mean, however, that transitions $e_1, \dots, e_k$ are executed one after another. Only transitions that are not independent must be executed in such manner (in order to guarantee their indivisibility). Independent transitions that are enabled can be executed also concurrently, and that does not change the result. This means that the order of such transitions is irrelevant and can be reflected by considering partially ordered analogons of firing sequences (cf. the concepts of a process in [3], [4], [5], and [8]). This also means that a marking not necessarily represents a global state of execution at a certain moment (if independent transitions are executed concurrently then there may be no moment at which such global state is defined). It rather represents a collection of local states which could potentially hold valid simultaneously, but actually may hold valid in different or even disjoint time intervals.

A marking m′ is said to be <u>reachable</u> from m if there exists a firing sequence x such that m′=mx.

A marking m′ is said to be <u>dead</u> if no transition is fireable under m′.

The entire execution starting from m is represented by a string $y=e_1e_2\ldots$ such that each finite initial segment of y is a firing sequence and y is finite if and only if my is a dead marking.

Since the order of independent transitions is irrelevant, different strings as described may represent the same execution. On the other hand, since possible conflicts between transitions which are not independent can be solved in different ways, there may be essentially different executions starting from the same marking.

In order to describe our implementation of Petri nets, we extend the concept of Petri nets by assuming that tokens are records carrying information, and that such information is processed during executing transitions. It is enough to do it for condition/event nets. The corresponding nets, called interpreted ones, can be defined as follows.

An <u>interpreted net</u> is Q=(P,I), where P=(B,E,F) is a condition/event net and I is a function, called <u>interpretation</u>, which assigns to each place the data carried by every token residing in this place and to each transition the transformation performed during executing the transition of the data from input places into data in output places (see sections 4,5, and 6, for examples).

A marking of such net can be regarded as a family $m=(m(b): b\in B)$ of sets of tokens residing in the corresponding places, where each $m(b)$ contains a single token, denoted <u>token</u>$(m(b))$, or is empty (<u>token</u>$(m(b))$ is not defined). If $I(b)=(i,j,\dots)$ for a place b then <u>token</u>$(m(b))$, when defined, is regarded to contain data denoted respectively $i($<u>token</u>$(m(b)))$, $j($<u>token</u>$(m(b)))$,... .

Fireability of a transition e under such marking means that $m(b)\neq\emptyset$ for all $b\in Fe$ and $m(b)=\emptyset$ for all $b\in eF-Fe$. The execution of e changes m to me as follows:

$$me(b)=\begin{cases}\emptyset \text{ for } b\in Fe-eF\\ \{\underline{b\text{-token}}\} \text{ for } b\in eF-Fe\\ m(b) \text{ for other } b\,,\end{cases}$$

where <u>b-token</u> is a suitable token which may be carried by b. The information carried by the tokens of m are assumed to be processed into that carried by the tokens of me as specified by I(e) (see sections 4,5, and 6).

In our description each condition/event net $P=(B,E,F)$ is considered together with an invariant subset C of markings. The pair (P,C) (or the quadruple (B,E,F,C)) is usually called a <u>condition/event system</u> (cf. Petri [6]) or simply a <u>system</u>. In case of interpreted net $Q=(P,I)$ we have to do with an <u>interpreted system</u> (Q,C).

The paper is organized as follows.

In section 2 we describe the modules and the construction of the system which is supposed to execute a Petri net.

The general idea (based on [10]) of the function of the system is described in section 3.

A detailed description of basic modules as interpreted systems is given in sections 4 and 5. The interpreted systems presented there are combined in section 6 into one system which describes the implementation of entire Petri net.

In section 7 we formulate properties which have to be proved in order to show correctness of implementation.

The corresponding proofs are given in section 8.

The paper ends with final remarks that are collected in section 9.

It should be mentioned that some attempts of implementing Petri nets by interconnected modules have already been made (cf. Furtek [1] and Priese [7], for example). Our problem statement and the idea of solving it are, however, different from those known from other works.

2. The construction

A Petri net P=(B,E,F) will be executed by modules assigned to places and transitions, the modules corresponding to adjacent nodes connected by communication lines.

To each place b (resp.: transition e) we assign a module (a kind of automaton), denoted $\underline{\text{control}}_b$ (resp.: $\underline{\text{control}}_e$), which plays the role of local sequential control of b (resp.: of e). If b is an input or output place of e (i.e., bFe or eFb) then we establish a communication line from $\underline{\text{control}}_b$ to $\underline{\text{control}}_e$, denoted $\underline{\text{line}}_{be}$, and a communication line from $\underline{\text{control}}_e$ to $\underline{\text{control}}_b$, denoted $\underline{\text{line}}_{eb}$. Finally, with the aid of a successor function $\underline{\text{next}}_x$, we introduce a circular order of all incoming and outgoing lines of each $\underline{\text{control}}_x$ ($q=\underline{\text{next}}_x(p)$ means that q follows immediately p).

Each $\underline{\text{line}}_{xy}$ serves to send information from $\underline{\text{control}}_x$ (the sender) to $\underline{\text{control}}_y$ (the receiver). Such line is supposed to work as a register which may be loaded by the sender when empty, and emptied by the receiver when loaded. The sender sends information to the receiver by loading it into the line when the line is empty (a write operation). The receiver accepts the information and empties the line when the information is loaded into the line (a read operation). This mode of communication prevents the sender and the receiver from simultaneously using the line and from a loss of information.

The local sequential controls are supposed to scan their incoming and outgoing communication lines (according to the introduced circular orders) and react properly. If an incoming line is scanned which currently is loaded then the control reads the contents of the line (by which the line is emptied), process it, and goes to scanning the next line. If such line is empty then the control goes immediately to scanning the next line. If an outgoing line is scanned which currently is empty then the control works out the information to be sent, loads it into the line, and goes to scanning the next line. If such line is loaded then the control goes immediately to scanning the next line.

Working in such manner, the local sequential controls are able to exchange information by the established communication lines and store it. In particular, the control of a transition e is able to follow (possibly with a delay) the situations in all adjacent places (the control of each adjacent place b can report the situation in b to the control of e) and deposit certain information in such places or remove it, if necessary (by sending suitable signals to the controls of the corresponding places).

Observe that each local sequential control (of a place or a transition) is a module which reacts only on its own state and on the state of its communication lines.

The same remains true for combinations as shown in fig. 1 of local sequential controls and connecting them communication lines. Each combination of this kind can also be regarded as a module. The state of such module consists of the states of its components (modules and internal communication lines). The communication lines of the module are those connecting it with its environment.

In consequence, every subset of transitions is implemented by the module consisting of the controls of these transitions, the controls of their adjacent places, and the communication lines which connect the above components.

Observe that such module depends only on the particular subset of transitions (taken together with the necessary context of adjacent places), and it does not depend on entire net.

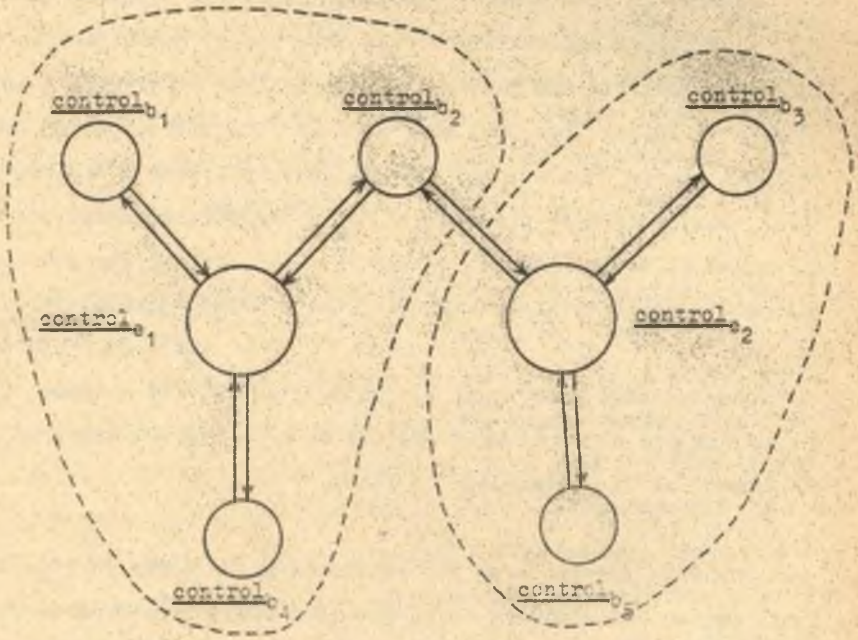

Fig. 1

3. The function of construction

The implemented net is executed by the controls of transitions working permanently and co-operating with the controls of adjacent places. The control of each transition follows the situations in adjacent places and tries to execute the transition whenever this transition is enabled. When several transitions with common adjacent places are enabled a conflict arises which must be solved by the controls of transitions. In order to solve conflicts the controls of enabled transitions play a game and only the control that wins is allowed to execute the transition it is assigned to (cf. [10]).

The game is played with the aid of local lists placed in places, called local priorities, which jointly represent a global priority among the controls of transitions. These lists consist of (names of) adjacent transitions. The control of a transition e is regarded to be of higher priority than that of e if e precedes e in a local list. The local lists must be consistent in the sense that the relation "to be of higher priority" must be a partial order. For simplicity the local lists are supposed to be static (in [10] dynamic local priorities have been considered).

The game is played permanently with the varying set of controls of enabled transitions as players.

Depending on the current situation the players are distributing their visiting-cards in the adjacent places of the corresponding transitions (one card of a player in a place) or collecting the already distributed visiting-cards back. The visiting-cards which are present in a place are arranged into a queue. A player wins if he succeeds to distribute his visiting-cards in all adjacent places of the transition he is assigned to such that the cards are first ones in all the corresponding queues. Such a player announces his winning in all adjacent places of the transition he is assigned to.

The rules of the game are as follows.

A player may start distributing his visiting-cards in adjacent places of the transition he is assigned to and continue this process while the following three conditions are fulfilled:

(d1) the transition is enabled,

(d2) in adjacent places there are no announcements of winning of other players,

(d3) in adjacent places there are no visiting-cards of players of higher priorities or such cards are preceded by a visiting-card of the player.

A player is obliged to stop distributing his visiting-cards and collect back the cards he has already distributed whenever one of the following three events occurs:

(r1) the transition the player is assigned to ceases being enabled,

(r2) in an adjacent place an announcement of winning of another player appears,

(r3) in an adjacent place a visiting-card of a player of higher priority appears to precede the card the player has left or is going to leave in the queue.

The winner announces his winning in all adjacent places of the transition he is assigned to and next he removes his own cards from such places and waits until cards of other players are also removed. Then he changes the marking of adjacent places according to the firing rule. Finally, the winner removes his announcements of winning from all adjacent places. This ends the current execution of the transition.

4. The behaviour of the control of a place

The control of a place b has incoming lines from and outgoing lines to the controls of all adjacent transitions $e \in Fb \cup bF$. It maintains information t_b on the current situation in b. This information is updated according to the signals received from the controls of adjacent transitions and reported to these controls. In order to read signals and report information the control of b scans the incoming and outgoing lines in the circular order given by a successor function $\underline{next}_b$ and performs read and write operations when possible. In what follows we assume that $\underline{line}_{be} = \underline{next}_b(\underline{line}_{eb})$ for all $e \in Fb \cup bF$ and by $\underline{priority}(b)$ we denote the list of (names of controls of) adjacent transitions $e \in Fb \cup bF$ (each e occurring exactly once) which represents the local priority in b (e is regarded to be of higher priority than e' if e precedes e' in the list).

The information t_b that $\underline{control}_b$ maintains consists of the following data:

$\underline{queue}(t_b)$: a list of (visiting-cards of controls of) adjacent transitions $e \in Fb \cup bF$, each e occurring at most once, winners(t_b): a subset of (names of controls of) adjacent transitions $e \in Fb \cup bF$ which contains at most one element,

marking(t_b): a non-negative integer (the current number of tokens in b), where marking(t_b) $\leq$ capacity(b).

By T_b we denote the set of all t_b of this kind.

From the behavioural point of view $control_b$ can be regarded as an interpreted system $S_b^* = (Q_b^*, C_b^*)$, where $Q_b^* = (P_b^*, I_b^*)$ is an interpreted net, i.e., a net $P_b^* = (B_b^*, E_b^*, F_b^*)$ with an interpretation I_b^*, and C_b^* is a set of markings of P_b^*. We define B_b^* as the set of places $at_b(line_{eb})$, $at_b(line_{be})$, loaded($line_{eb}$), empty($line_{eb}$), loaded($line_{be}$), empty($line_{be}$) ($e \in Fb \cup bF$), and E_b^* as the set of transitions $read_b(line_{eb})$, $skip_b(line_{eb})$, $write_b(line_{be})$, $skip_b(line_{be})$ ($e \in Fb \cup bF$). The relation F_b^* and interpretation I_b^* are defined as shown in fig. 2 (the information a token being in a place contains is specified at the corresponding circle; how the information contained in tokens is processed during executing transitions is specified in the corresponding boxes). C_b^* is the set of markings such that: at most one of all places $at_b(line_{eb})$, $at_b(line_{be})$ ($e \in Fb \cup bF$) carries a token, at most one of each two places loaded($line_{eb}$), empty($line_{eb}$) ($e \in Fb \cup bF$) carries a token, and at most one of each two places loaded($line_{be}$), empty($line_{be}$) ($e \in Fb \cup bF$) carries a token.

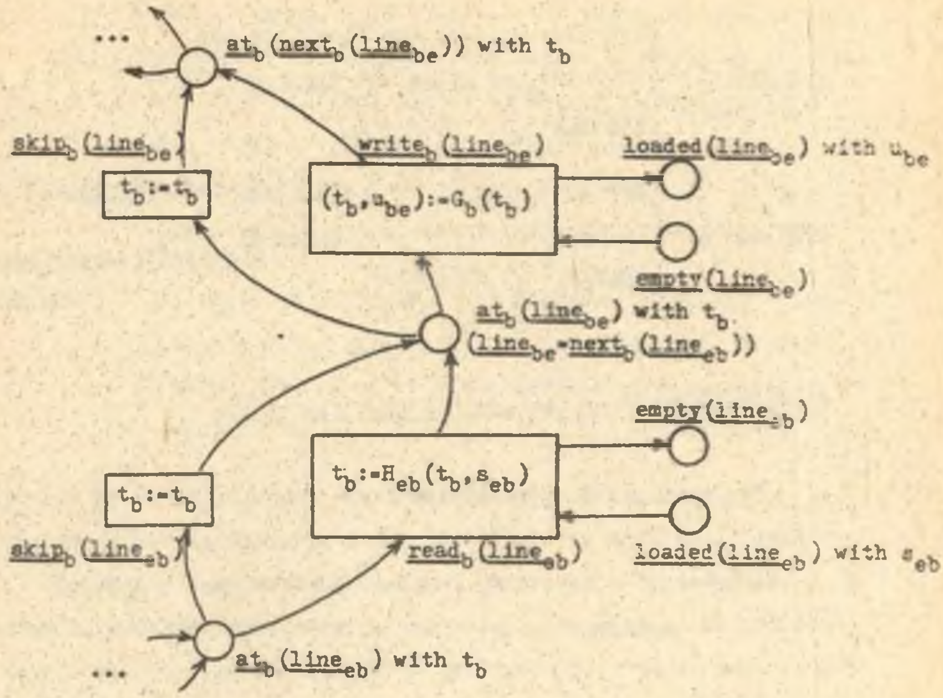

Fig. 2

The functions G_b and H_{eb} are defined as follows:

$G_b(t_b)=(t_b,t_b)$ for all $t_b \in T_b$,

$H_{eb}(t_b,s_{eb})=t'_b$,

where:

$$\underline{queue}(t'_b)=\begin{cases}\underline{queue}(t_b)e \text{ when } s_{eb}=\underline{card} \text{ and } e \text{ does not occur in } \underline{queue}(t_b)\\ xy \text{ when } s_{eb}=\underline{cardback} \text{ and } \underline{queue}(t_b)=xey\\ \underline{queue}(t_b) \text{ otherwise,}\end{cases}$$

$$\underline{\text{winners}}(t'_b)=\begin{cases}\{e\} \text{ when } s_{eb}=\underline{\text{won}} \text{ and } \underline{\text{winners}}(t_b)=\emptyset\\ \emptyset \text{ when } s_{eb}=\underline{\text{release}} \text{ and } \underline{\text{winners}}(t_b)=\{e\}\\ \underline{\text{winners}}(t_b) \text{ otherwise,}\end{cases}$$

$$\underline{\text{marking}}(t'_b)=\begin{cases}\underline{\text{marking}}(t_b)-1 \text{ when } s_{eb}=\underline{\text{decrease}} \text{ and } \underline{\text{marking}}(t_b)>0\\ \underline{\text{marking}}(t_b)+1 \text{ when } s_{eb}=\underline{\text{increase}} \text{ and } \underline{\text{marking}}(t_b)<\underline{\text{capacity}}(b)\\ \underline{\text{marking}}(t_b) \text{ otherwise.}\end{cases}$$

5. The behaviour of the control of a transition

The control of a transition e has outgoing lines to and incoming lines from the controls of all adjacent places $b \in {}^{\bullet}e \cup e^{\bullet}$. It maintains information v_e consisting of current images of situations in adjacent places and of some local data. This information is updated according to reports on current situations received from the controls of adjacent places and it serves the control of e to decide what signals should be sent to the controls of adjacent places. In order to send signals to the controls of adjacent places and read reports from the controls of adjacent places the control of e scans the outgoing and incoming lines in the circular order given by a successor function $\underline{\text{next}}_e$ and performs write and read operations when possible. In what follows we assume that $\underline{\text{line}}_{be}=\underline{\text{next}}_e(\underline{\text{line}}_{eb})$ for all $b \in {}^{\bullet}e \cup e^{\bullet}$.

The information v_e that $\underline{\text{control}}_e$ maintains consists of the following data:

$\underline{\text{image}}(v_e)$: a family $(\underline{\text{image}}_b(v_e)$: $b \in {}^{\bullet}e \cup e^{\bullet})$ of current images situations in adjacent places $b \in {}^{\bullet}e \cup e^{\bullet}$, sent($v_e$): a family ($\underline{\text{sent}}_b(v_e)$: $b \in Fe \cup eF$) of sets of signals which have most recently been sent to the controls of the corresponding adjacent places $b \in Fe \cup eF$, each $\underline{\text{sent}}_b(v_e)$ being empty or containing a single signal - the last one which has been sent to $\underline{\text{control}}_b$ but possibly not yet taken into account,

updated(v_e): a family ($\underline{\text{updated}}_b(v_e)$: $b \in Fe \cup eF$) of boolean values ($\underline{\text{updated}}_b(v_e)$=true represents the fact that during current execution of e the marking of b has already been updated),

phase(v_e): a value characterizing the current activity of $\underline{\text{control}}_e$.

The control of a transition is assumed to be able to send the following signals to the controls of adjacent places:

card,
cardback,
won,
decrease,
increase,
release,
none.

The control of a transition is assumed to be in one of the following phases of activity:

waiting,
distributing,
removing,
winning,

cleaning.

accessing.

releasing.

From the behavioural point of view $\underline{\text{control}}_e$ can be regarded as an interpreted system $S^*_e=(Q^*_e,C^*_e)$, where $Q^*_e=(P^*_e,I^*_e)$ is an interpreted net with the underlying net $P^*_e=(B^*_e,E^*_e,F^*_e)$ and interpretation I^*_e, and C^*_e is a set of markings of P^*_e. We define B^*_e as the set of places $\underline{\text{at}}_e(\underline{\text{line}}_{eb})$, $\underline{\text{at}}_e(\underline{\text{line}}_{be})$, $\underline{\text{loaded}}(\underline{\text{line}}_{eb})$, $\underline{\text{empty}}(\underline{\text{line}}_{eb})$, $\underline{\text{loaded}}(\underline{\text{line}}_{be})$, $\underline{\text{empty}}(\underline{\text{line}}_{be})$ ($b \in Fe \cup eF$), and E^*_e as the set of $\underline{\text{write}}_e(\underline{\text{line}}_{eb})$, $\underline{\text{skip}}_e(\underline{\text{line}}_{eb})$, $\underline{\text{read}}_e(\underline{\text{line}}_{be})$, $\underline{\text{skip}}_e(\underline{\text{line}}_{be})$ ($b \in Fe \cup eF$). The relation F^*_e and interpretation I^*_e are defined as shown in fig. 3. C^*_e is defined as the set of markings such that: at most one of all places $\underline{\text{at}}_e(\underline{\text{line}}_{eb})$, $\underline{\text{at}}_e(\underline{\text{line}}_{be})$ ($b \in Fe \cup eF$) carries a token, at most one of each two places $\underline{\text{loaded}}(\underline{\text{line}}_{eb})$, $\underline{\text{empty}}(\underline{\text{line}}_{eb})$ ($b \in Fe \cup eF$) carries a token, and at most one of each two places $\underline{\text{loaded}}(\underline{\text{line}}_{be})$, $\underline{\text{empty}}(\underline{\text{line}}_{be})$ ($b \in Fe \cup eF$) carries a token.

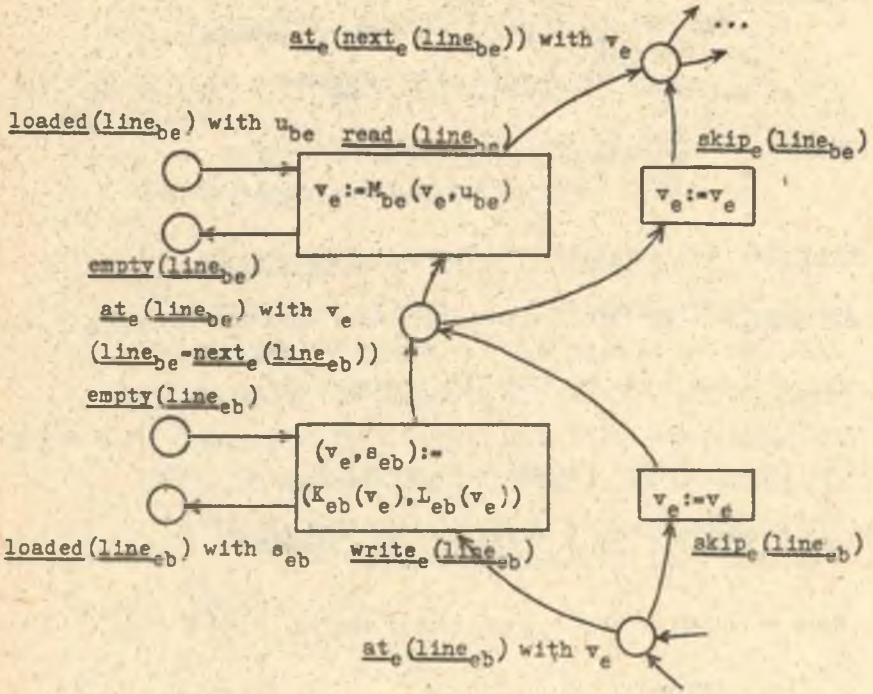

Fig. 3

The functions K_{eb}, L_{eb}, and M_{be}, are defined as follows.

For a family $w = (w_b\colon b \in Fe \cup eF)$ of possible (images of) situations $w_b \in T_b$ in adjacent places $b \in Fe \cup eF$ we define the following predicates:

$\underline{enabled}_e(w)$: $\Leftrightarrow$ $(\forall b\colon b \in Fe)(0 < \underline{marking}(w_b))$
and $(\forall b\colon b \in eF - Fe)(\underline{marking}(w_b) < \underline{capacity}(b))$.

$\underline{open}_e(w)$: $\Leftrightarrow$ $\underline{enabled}_e(w)$ and $(\forall b\colon b \in Fe \cup eF)((\underline{winners}(w_b) = \emptyset)$ and
$(\forall e'\colon e'$ precedes e in $\underline{queue}(w_b))(e$ precedes e'
in $\underline{priority}(b)))$.

$\underline{\text{shouldremove}}_e(w)$: $\Leftrightarrow$ not $\underline{\text{open}}_e(w)$ or
$(\exists b: b \in Fe \cup eF)(\exists e': e'$ precedes e in $\underline{\text{queue}}(w_b))(e'$ precedes e in $\underline{\text{priority}}(b))$.

$\underline{\text{infront}}_e(w)$: $\Leftrightarrow$ $\underline{\text{open}}_e(w)$ and
$(\forall b: b \in Fe \cup eF)(e$ is first in $\underline{\text{queue}}(w_b))$.

$\underline{\text{announced}}_e(w)$: $\Leftrightarrow$ $(\forall b: b \in Fe \cup eF)(\underline{\text{winners}}(w_b)=\{e\})$.

$\underline{\text{cardsremoved}}_e(w)$: $\Leftrightarrow$ $(\forall b: b \in Fe \cup eF)(e$ does not occur in $\underline{\text{queue}}(w_b))$.

$\underline{\text{released}}_e(w)$: $\Leftrightarrow$ $(\forall b: b \in Fe \cup eF)(\underline{\text{winners}}(w_b) \neq \{e\})$.

Besides, for v_e which is maintained by $\underline{\text{control}}_e$, we define:

$\underline{\text{fired}}_e(v_e)$: $\Leftrightarrow$ $(\forall b: b \in (Fe-eF) \cup (eF-Fe))$ $\underline{\text{updated}}_b(v_e)$.

Then we consider $w=(w_b: b \in Fe \cup eF)$, where:

$$w_p = \begin{cases} u_{be} & \text{for } p=b \\ \underline{\text{image}}_p(v_e) & \text{for other } p \in Fe \cup eF, \end{cases}$$

and define $v'_e = M_{be}(v_e, u_{be}))$ in the following manner:

$\underline{\text{image}}(v'_e)=w$,

$\underline{\text{sent}}_p(v'_e)=\underline{\text{sent}}_p(v_e)$ for $p \in Fe \cup eF$ and $p \neq b$,

$$\underline{\text{sent}}_b(v'_e)=\begin{cases} \emptyset \text{ when } \underline{\text{sent}}_b(v_e)=\{\underline{\text{card}}\} \text{ and } e \text{ occurs in } \underline{\text{queue}}(u_{be}) \\ \quad \text{or } \underline{\text{sent}}_b(v_e)=\{\underline{\text{cardback}}\} \text{ and } e \text{ does not occur in } \underline{\text{queue}}(u_{be}) \\ \quad \text{or } \underline{\text{sent}}_b(v_e)=\{\underline{\text{won}}\} \text{ and } \underline{\text{winners}}(u_{be})=\{e\} \\ \quad \text{or } \underline{\text{sent}}_b(v_e)=\{\underline{\text{decrease}}\} \text{ and } \underline{\text{marking}}(u_{be})= \\ \qquad\qquad \underline{\text{marking}}(\underline{\text{image}}_b(v_e))-1 \\ \quad \text{or } \underline{\text{sent}}_b(v_e)=\{\underline{\text{increase}}\} \text{ and } \underline{\text{marking}}(u_{be})= \\ \qquad\qquad \underline{\text{marking}}(\underline{\text{image}}_b(v_e))+1 \\ \quad \text{or } \underline{\text{sent}}_b(v_e)=\{\underline{\text{release}}\} \text{ and } \underline{\text{winners}}(u_{be}) \neq \{e\} \\ \underline{\text{sent}}_b(v_e) \text{ otherwise,} \end{cases}$$

$\underline{\text{updated}}_p(v'_e)=\underline{\text{updated}}_p(v_e)$ for $p \in Fe \cup eF$ and $p \neq b$,

$$\underline{\text{updated}}_b(v'_e)=\begin{cases}\underline{\text{true}} \text{ when } \underline{\text{sent}}_b(v_e)=\{\underline{\text{decrease}}\} \text{ and } \underline{\text{marking}}(u_{be})= \\ \qquad\qquad\qquad \underline{\text{marking}}(\underline{\text{image}}_b(v_e))-1 \\ \quad \text{or } \underline{\text{sent}}_b(v_e)=\{\underline{\text{increase}}\} \text{ and } \underline{\text{marking}}(u_{be})= \\ \qquad\qquad\qquad \underline{\text{marking}}(\underline{\text{image}}_b(v_e))+1 \\ \underline{\text{false}} \text{ when } \underline{\text{phase}}(v_e)\neq\underline{\text{accessing}} \\ \underline{\text{updated}}_b(v_e) \text{ otherwise,}\end{cases}$$

$$\underline{\text{phase}}(v'_e)=\begin{cases}\underline{\text{distributing}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{waiting}} \text{ and } \underline{\text{open}}_e(w) \\ \underline{\text{removing}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{distributing}} \text{ and } \underline{\text{shouldremove}}_e(w) \\ \underline{\text{winning}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{distributing}} \text{ and } \underline{\text{infront}}_e(w) \\ \underline{\text{cleaning}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{winning}} \text{ and } \underline{\text{announced}}_e(w) \\ \underline{\text{accessing}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{cleaning}} \text{ and } \underline{\text{cardsremoved}}_e(w) \\ \underline{\text{releasing}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{accessing}} \text{ and } \underline{\text{fired}}_e(v_e) \\ \underline{\text{waiting}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{releasing}} \text{ and } \underline{\text{released}}_e(w) \\ \quad \text{or } \underline{\text{phase}}(v_e)=\underline{\text{removing}} \text{ and } \underline{\text{cardsremoved}}_e(w) \\ \underline{\text{phase}}(v_e) \text{ otherwise.}\end{cases}$$

Thus we have defined the function M_{be}.

The function L_{eb} can be defined as follows:

$$L_{eb}(v_e)=\begin{cases}\underline{\text{card}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{distributing}} \text{ and } \underline{\text{sent}}_b(v_e)=\emptyset \text{ and} \\ \qquad e \text{ does not occur in } \underline{\text{queue}}(\underline{\text{image}}_b(v_e)) \\ \underline{\text{cardback}} \text{ when } \underline{\text{phase}}(v_e)\in\{\underline{\text{removing}},\underline{\text{cleaning}}\} \text{ and } \underline{\text{sent}}_b(v_e)=\emptyset \\ \qquad \text{and } e \text{ occurs in } \underline{\text{queue}}(\underline{\text{image}}_b(v_e)) \\ \underline{\text{won}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{winning}} \text{ and } \underline{\text{sent}}_b(v_e)=\emptyset \text{ and} \\ \qquad \underline{\text{winners}}(\underline{\text{image}}_b(v_e))=\emptyset \\ \underline{\text{decrease}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{accessing}} \text{ and } \underline{\text{sent}}_b(v_e)=\emptyset \text{ and} \\ \qquad b\in Fe-eF \text{ and not } \underline{\text{updated}}_b(v_e) \\ \underline{\text{increase}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{accessing}} \text{ and } \underline{\text{sent}}_b(v_e)=\emptyset \text{ and} \\ \qquad b\in eF-Fe \text{ and not } \underline{\text{updated}}_b(v_e) \\ \underline{\text{release}} \text{ when } \underline{\text{phase}}(v_e)=\underline{\text{releasing}} \text{ and } \underline{\text{sent}}_b(v_e)=\emptyset \text{ and} \\ \qquad \underline{\text{winners}}(\underline{\text{image}}_b(v_e))=\{e\} \\ \underline{\text{none}} \text{ otherwise.}\end{cases}$$

Finally, we define $v_e'=K_{eb}(v_e)$ in the following manner:

$\underline{image}_p(v_e')=\underline{image}_p(v_e)$ for all $p\in Fe\cup eF$,

$\underline{sent}_p(v_e')=\underline{sent}_p(v_e)$ for $p\in Fe\cup eF$ and $p\neq b$,

$$\underline{sent}_b(v_e')=\begin{cases}\{L_{eb}(v_e)\} \text{ when } \underline{sent}_b(v_e)=\emptyset \text{ and } L_{eb}(v_e)\neq \underline{none}\\ \underline{sent}_b(v_e) \text{ otherwise,}\end{cases}$$

$\underline{updated}_p(v_e')=\underline{updated}_p(v_e)$ for all $p\in Fe\cup eF$,

$\underline{phase}(v_e')=\underline{phase}(v_e)$.

6. The behaviour of entire construction

The construction consists of the controls of places and the controls of transitions, each two controls of elements that are adjacent connected by two communication lines as described in section 2. The information processed in this construction consists of what is processed in particular components and of the information existing in communication lines. The only operations are those of the modules controlling transitions and places. The modules are not synchronized from outside. What they actually do depends only on the information they exchange.

The behaviour of the entire construction can be described by the interpreted system whose net is the union of the nets of the components (as shown in fig. 4) and whose markings are the ones admissible for all components. Formally, such system is $S^*=(Q^*,C^*)$, where $Q^*=(P^*,I^*)$ is an interpreted net with the underlying net $P^*=(B^*,E^*,F^*)$ and interpretation I^*, and C^* is a set of markings of P^*. We define Q^* as $\bigcup(Q_x^* : x\in B\cup E)$, i.e., we assume that $B^*=\bigcup(B_x^*: x\in B\cup E)$, $E^*=\bigcup(E_x^*: x\in B\cup E)$, $F^*=\bigcup(F_x^*: x\in B\cup E)$

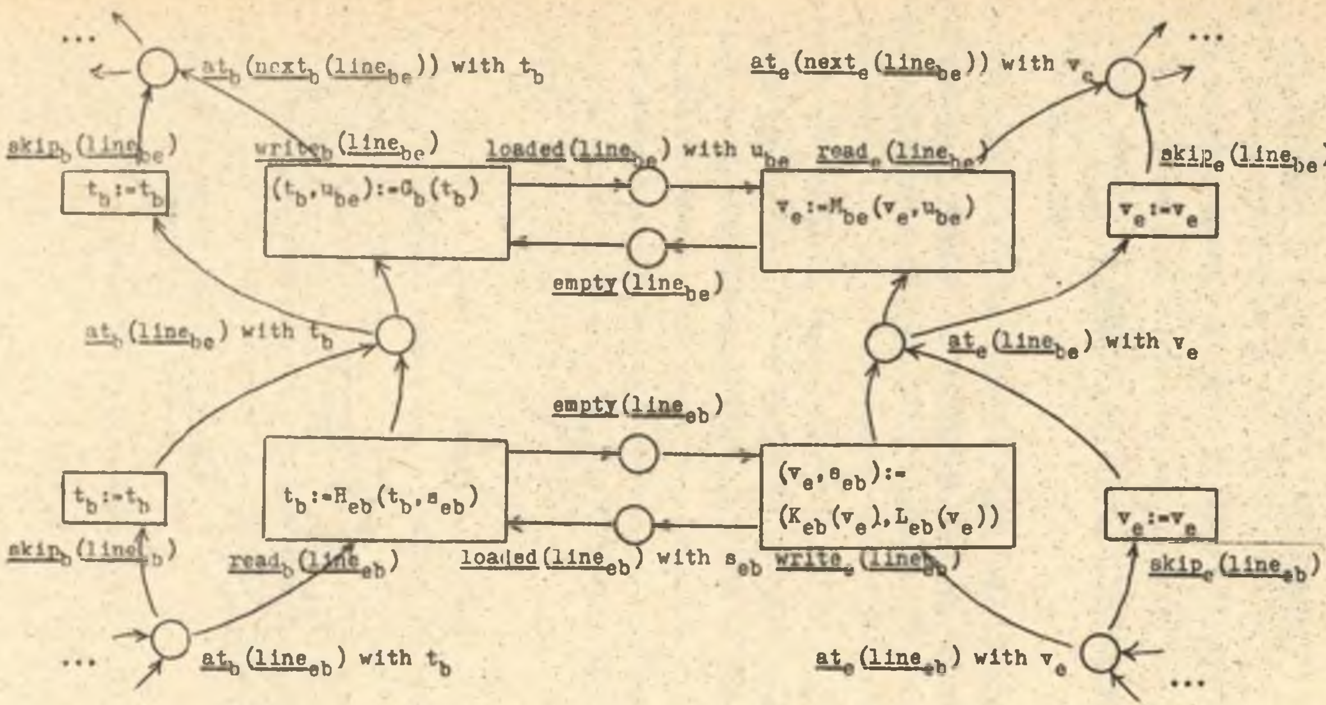
...
$at_b(next_b(line_{be}))$ with t_b
$at_e(next_e(line_{be}))$ with v_e
...
$skip_b(line_{be})$
$write_b(line_{be})$
$loaded(line_{be})$ with u_{be}
$read_e(line_{be})$
$skip_e(line_{be})$
$t_b:=t_b$
$(t_b,u_{be}):=G_b(t_b)$
$v_e:=M_{be}(v_e,u_{be})$
$v_e:=v_e$
$empty(line_{be})$
$at_b(line_{be})$ with t_b
$at_e(line_{be})$ with v_e
$empty(line_{eb})$
$t_b:=t_b$
$t_b:=H_{eb}(t_b,s_{eb})$
$(v_e,s_{eb}):=$
$(K_{eb}(v_e),L_{eb}(v_e))$
$v_e:=v_e$
$skip_b(line_{eb})$
$read_b(line_{eb})$
$loaded(line_{eb})$ with s_{eb}
$write_e(line_{eb})$
$skip_e(line_{eb})$
...
$at_b(line_{eb})$ with t_b
$at_e(line_{eb})$ with v_e
...

Fig. 4

(which means that $P^* = \bigcup(P_x^*: x \in B \cup E)$), and $I^* = \bigcup(I_x^*: x \in B \cup E)$. C^* is defined as the set of markings c such that $c|B_x^* \in C_x^*$ for all $x \in B \cup E$. Observe that all E_x^* are mutually disjoint, all P_x^* are mutually disjoint, B_x^* and B_y^* are disjoint if neither xFy nor yFx, and that

$$B_b^* \cap B_e^* = \{\underline{loaded}(\underline{line}_{eb}), \underline{empty}(\underline{line}_{eb}), \underline{loaded}(\underline{line}_{be}), \underline{empty}(\underline{line}_{be})\}$$

whenever bFe or eFb for a place b and transition e. A part of the entire interpreted net which describes the exchange of information between $\underline{control}_b$ and $\underline{control}_e$, where b is a place and e is a transition such that bFe or eFb, is shown in fig. 4. Such a part can be represented schematically as shown in fig. 5. Since the subnets describing singular components are of circular form, the interaction of components can be represented schematically as shown in fig. 6.

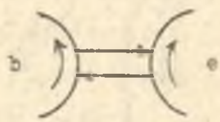

Fig. 5

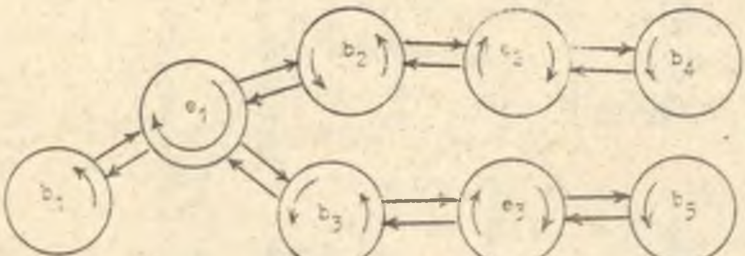

Fig. 6

7. <u>The problem of correctness of implementation</u>

The interpreted net $Q^{\ast}$ is an abstract description of the behaviour of modules. It does not say, however, how the execution of the given net P proceeds in time. Now we make some assumptions about that.

First of all, we assume that each module scans each of its communication lines in a finite time and that it is not prevented by anything from continuing this process.

Next, we assume that a module scanning a communication line skips this line only if it cannot execute the corresponding read or write operation. Besides, we assume that all modules work at comparable speeds in the sense that, given a module and its communication line, the number of transitions of $Q^{\ast}$ executed by modules before the given communication line is scanned by the given module is finite.

Finally, we assume that the time of execution of a transition of $Q^{\ast}$ by a module is random and varies from one execution to another.

The above assumptions can be expressed by restricting the set of formal executions of $Q^{\ast}$ to a subset of executions which may really happen, called real ones.

The first assumption is expressed by the following axiom.

Axiom 1. If $y=f_1f_2\ldots$ is a real execution of Q^* starting from a marking $n\in C^*$ then there are not any transition f and number k such that f is fireable under all markings $nf_1\ldots f_k$, $nf_1\ldots f_kf_{k+1},\ldots$ and f does not occur among $f_{k+1},f_{k+2},\ldots$.

The next two assumptions are expressed by the following axiom.

Axiom 2. If $y=f_1f_2\ldots$ is a real execution of Q^* starting from a marking $n\in C^*$ and f is a read or write transition of a module such that f is enabled under a marking $nf_1\ldots f_k$ then f occurs among $f_{k+1},f_{k+2},\ldots$.

Finally, the last assumption is expressed as follows.

Axiom 3. All formal executions of Q^* which satisfy axioms 1 and 2 are real executions of Q^*.

That the system of modules we have constructed executes the given Petri net can be formulated as follows.

For every marking m of the given net P, by executions(P,m) we denote the set of all executions of P starting from m. Similarly, for every marking $n\in C^*$ of the interpreted net Q^*, by executions(Q^*,n) we denote the set of all formal executions of Q^* starting from n. Finally, by realexecutions(Q^*,n) we denote the subset of real executions of Q^* starting from $n\in C^*$.

Then we require defining a subset $D^*\subseteq C^*$ of markings of Q^* and two functions U and W such that:

(1) D^{*} is invariant under all transitions of Q^{*},

(2) to every $n \in D^{*}$ there corresponds a marking $U(n)$ of P,

(3) every marking of P is of the form $U(n)$ for some $n \in D^{*}$,

(4) to every string y of transitions of Q^{*} that is a real execution of Q^{*} starting from $n \in D^{*}$, or an initial segment of such execution, there corresponds a string $W(y)$ of transitions of P,

(5) $W(yz)=W(y)W(z)$ whenever $W(yz)$ is defined,

(6) $U(n)W(y)=U(ny)$ whenever $n \in D^{*}$ and ny is defined,

(7) $W(y) \in \underline{\text{executions}}(P,U(n))$ whenever $n \in D^{*}$ and $y \in \underline{\text{realexecutions}}(Q^{*},n)$.

(8) every string x of transitions of P that is a finite execution of P starting from a marking m, or a finite initial segment of an infinite execution starting from m, is of the form $W(y)$ for a real execution y of Q^{*} starting from $n \in D^{*}$ such that $U(n)=m$, or, respectively, for an initial segment of such execution.

The possibility of defining such subset D^{*} and functions U and W means that all firing sequences of P can be realized by the system of modules, and that all runs of the system of modules starting from suitable markings can be regarded as executions of P. This is just what we have in mind speaking of correctness of implementation.

In order to define D^{*}, U, and W, as required we consider markings $n \in C^{*}$ such that:

(i1) for every $b \in B$, exactly one of the places $\underline{at}_b(\underline{line}_{eb})$ and $\underline{at}_b(\underline{line}_{be})$ $(e \in Fb \cup bF)$ carries a token, denoted $t_b(n)$,

(i2) for every $e \in E$, exactly one of the places $\underline{at}_e(\underline{line}_{eb})$ and $\underline{at}_e(\underline{line}_{be})$ $(b \in Fe \cup eF)$ carries a token, denoted $v_e(n)$,

(i3) for every $(b,e) \in F$ and every $(e,b) \in F$, exactly one of the two places $\underline{loaded}(\underline{line}_{be})$, $\underline{empty}(\underline{line}_{be})$ carries a token denoted respectively $u_{be}(n)$ or $r_{be}(n)$ when defined, and exactly one of the two places $\underline{loaded}(\underline{line}_{eb})$, $\underline{empty}(\underline{line}_{eb})$ carries a token denoted respectively $s_{eb}(n)$ or $r_{eb}(n)$ when defined,

(i4) for every $e \in E$, if $\underline{phase}(v_e(n))=\underline{waiting}$ then not $\underline{open}_e(\underline{image}(v_e(n)))$ and, for all $b \in Fe \cup eF$: $\underline{winners}(t_b(n)) \neq \{e\}$ and $\underline{winners}(\underline{image}_b(v_e(n))) \neq \{e\}$ and e does not occur in $\underline{queue}(t_b(n))$ and in $\underline{queue}(\underline{image}_b(v_e(n)))$ and not $\underline{updated}_b(v_e(n))$ and $\underline{sent}_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{none}$.

(i5) for every $e \in E$, if $\underline{phase}(v_e(n))=\underline{distributing}$ then $\underline{open}_e(\underline{image}(v_e(n)))$ and e does not occur or is not first in $\underline{queue}(\underline{image}_b(v_e(n)))$ for some $b \in Fe \cup eF$, and, for all $b \in Fe \cup eF$: not $\underline{updated}_b(v_e(n))$ and exactly one of the following conditions is fulfilled:

- e occurs in $\underline{queue}(t_b(n))$ and in $\underline{queue}(\underline{image}_b(v_e(n)))$ and $\underline{sent}_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{none}$,
- e occurs in $\underline{queue}(t_b(n))$ but not in $\underline{queue}(\underline{image}_b(v_e(n)))$ and $\underline{sent}_b(v_e(n))=\{\underline{card}\}$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{none}$,

- e does not occur in queue($t_b(n)$) and in queue(image$_b(v_e(n))$) and sent$_b(v_e(n))$=$\{$card$\}$ and $s_{eb}(n)$=card,

- e does not occur in queue($t_b(n)$) and in queue(image$_b(v_e(n))$) and sent$_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)$=none.

(i6) for every $e \in E$, if phase($v_e(n)$)=removing then e occurs in queue(image$_b(v_e(n))$) for some $b \in Fe \cup eF$ and, for all $b \in Fe \cup eF$: winners($t_b(n)$)$\neq\{e\}$ and winners(image$_b(v_e(n))$)$\neq\{e\}$ and not updated$_b(v_e(n))$ and exactly one of the following conditions is fulfilled:

- e does not occur in queue($t_b(n)$) and in queue(image$_b(v_e(n))$) and sent$_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)$=none,

- e does not occur in queue($t_b(n)$) but it occurs in queue(image$_b(v_e(n))$) and sent$_b(v_e(n))$=$\{$cardback$\}$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)$=none.

- e occurs in queue($t_b(n)$) and in queue(image$_b(v_e(n))$) and sent$_b(v_e(n))$=$\{$cardback$\}$ and $s_{eb}(n)$=cardback.

- e occurs in queue($t_b(n)$) and in queue(image$_b(v_e(n))$) and sent$_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)$=none.

(i7) for every $e \in E$, if phase($v_e(n)$)=winning then enabled$_e$(image($v_e(n)$)) and winners(image$_b(v_e(n))$)$\neq\{e\}$ for some $b \in Fe \cup eF$ and, for all $b \in Fe \cup eF$: e is first in queue($t_b(n)$) and in queue(image$_b(v_e(n))$) and not updated$_b(v_e(n))$ and exactly one of the following conditions is fulfilled:

- $\underline{\text{winners}}(t_b(n))=\underline{\text{winners}}(\underline{\text{image}}_b(v_e(n)))=\{e\}$ and $\underline{\text{sent}}_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$,

- $\underline{\text{winners}}(t_b(n))=\{e\}\neq\underline{\text{winners}}(\underline{\text{image}}_b(v_e(n)))$ and $\underline{\text{sent}}_b(v_e(n))=\{\underline{\text{won}}\}$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$,

- $\underline{\text{winners}}(t_b(n))\neq\{e\}\neq\underline{\text{winners}}(\underline{\text{image}}_b(v_e(n)))$ and $\underline{\text{sent}}_b(v_e(n))=\{\underline{\text{won}}\}$ and $s_{eb}(n)=\underline{\text{won}}$,

- $\underline{\text{winners}}(t_b(n))\neq\{e\}\neq\underline{\text{winners}}(\underline{\text{image}}_b(v_e(n)))$ and $\underline{\text{sent}}_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$,

(18) for every $e\in E$, if $\underline{\text{phase}}(v_e(n))=\underline{\text{cleaning}}$ then $\underline{\text{enabled}}_e(\underline{\text{image}}(v_e(n)))$ and e occurs in $\underline{\text{queue}}(\underline{\text{image}}_b(v_e(n)))$ for some $b\in Fe\cup eF$ and, for all $b\in Fe\cup eF$: $\underline{\text{winners}}(t_b(n))=\underline{\text{winners}}(\underline{\text{image}}_b(v_e(n)))=\{e\}$ and not $\underline{\text{updated}}_b(v_e(n))$ and exactly one of the following conditions is fulfilled:

- e does not occur in $\underline{\text{queue}}(t_b(n))$ and in $\underline{\text{queue}}(\underline{\text{image}}_b(v_e(n)))$ and $\underline{\text{sent}}_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$,

- e does not occur in $\underline{\text{queue}}(t_b(n))$ but it occurs in $\underline{\text{queue}}(\underline{\text{image}}_b(v_e(n)))$ and $\underline{\text{sent}}_b(v_e(n))=\{\underline{\text{cardback}}\}$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$,

- e occurs in $\underline{\text{queue}}(t_b(n))$ and in $\underline{\text{queue}}(\underline{\text{image}}_b(v_e(n)))$ and $\underline{\text{sent}}_b(v_e(n))=\{\underline{\text{cardback}}\}$ and $s_{eb}(n)=\underline{\text{cardback}}$,

- e occurs in $\underline{\text{queue}}(t_b(n))$ and in $\underline{\text{queue}}(\underline{\text{image}}_b(v_e(n)))$ and $\underline{\text{sent}}_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$.

(19) for every $e \in E$, if $\underline{\text{phase}}(v_e(n))=\underline{\text{accessing}}$ then not $\underline{\text{fired}}_e(v_e(n))$ and, for all $b \in Fe \cup eF$: $\underline{\text{winners}}(t_b(n))=\underline{\text{winners}}(\underline{\text{image}}_b(v_e(n)))=\{e\}$ and $\underline{\text{queue}}(t_b(n))$ is empty and $\underline{\text{queue}}(\underline{\text{image}}_b(v_e(n)))$ is empty and exactly one of the following conditions is fulfilled:

- $b \in Fe \cap eF$ and $\underline{\text{marking}}(t_b(n))=\underline{\text{marking}}(\underline{\text{image}}_b(v_e(n)))$ and not $\underline{\text{updated}}_b(v_e(n))$ and $\underline{\text{sent}}_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$.

- $b \in Fe-eF$ and $\underline{\text{updated}}_b(v_e(n))$ and $\underline{\text{marking}}(t_b(n))=\underline{\text{marking}}(\underline{\text{image}}_b(v_e(n))) \geqslant 0$ and $\underline{\text{sent}}_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$.

- $b \in Fe-eF$ and not $\underline{\text{updated}}_b(v_e(n))$ and $\underline{\text{marking}}(t_b(n))=\underline{\text{marking}}(\underline{\text{image}}_b(v_e(n)))-1 \geqslant 0$ and $\underline{\text{sent}}_b(v_e(n))=\{\underline{\text{decrease}}\}$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$.

- $b \in Fe-eF$ and not $\underline{\text{updated}}_b(v_e(n))$ and $\underline{\text{marking}}(t_b(n))=\underline{\text{marking}}(\underline{\text{image}}_b(v_e(n)))>0$ and $\underline{\text{sent}}_b(v_e(n))=\{\underline{\text{decrease}}\}$ and $s_{eb}(n)=\underline{\text{decrease}}$.

- $b \in Fe-eF$ and not $\underline{\text{updated}}_b(v_e(n))$ and $\underline{\text{marking}}(t_b(n))=\underline{\text{marking}}(\underline{\text{image}}_b(v_e(n)))>0$ and $\underline{\text{sent}}_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$.

- $b \in eF-Fe$ and $\underline{\text{updated}}_b(v_e(n))$ and $\underline{\text{marking}}(t_b(n))=\underline{\text{marking}}(\underline{\text{image}}_b(v_e(n))) \leqslant \underline{\text{capacity}}(b)$ and $\underline{\text{sent}}_b(v_e(n))=\emptyset$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$.

- $b \in eF-Fe$ and not $\underline{\text{updated}}_b(v_e(n))$ and $\underline{\text{marking}}(t_b(n))=\underline{\text{marking}}(\underline{\text{image}}_b(v_e(n)))+1 \leqslant \underline{\text{capacity}}(b)$ and $\underline{\text{sent}}_b(v_e(n))=\{\underline{\text{increase}}\}$ and either $r_{eb}(n)$ is defined or $s_{eb}(n)=\underline{\text{none}}$.

- $b \in eF - Fe$ and not updated$_b$(v_e(n)) and marking(t_b(n))=marking(image$_b$(v_e(n)))< capacity(b) and sent$_b$(v_e(n))={increase} and s_{eb}(n)=increase.

- $b \in eF - Fe$ and not updated$_b$(v_e(n)) and marking(t_b(n))=marking(image$_b$(v_e(n)))< capacity(b) and sent$_b$(v_e(n))=$\emptyset$ and either r_{eb}(n) is defined or s_{eb}(n)=none.

(i10) for every $e \in E$, if phase(v_e(n))=releasing then winners(image$_b$(v_e(n)))={e} for some $b \in Fe \cup eF$ and, for all $b \in Fe \cup eF$: e does not occur in queue(t_b(n)) and in queue(image$_b$(v_e(n))) and exactly one of the following conditions is fulfilled:

- winners(t_b(n))$\neq${e}$\neq$winners(image$_b$(v_e(n))) and not updated$_b$(v_e(n)) and sent$_b$(v_e(n))=$\emptyset$ and either r_{eb}(n) is defined or s_{eb}(n)=none.

- winners(t_b(n))$\neq${e}=winners(image$_b$(v_e(n))) and not updated$_b$(v_e(n)) and sent$_b$(v_e(n))={release} and either r_{eb}(n) is defined or s_{eb}(n)=none.

- winners(t_b(n))={e}=winners(image$_b$(v_e(n))) and queue(t_b(n)) and queue(image$_b$(v_e(n))) are empty and not updated$_b$(v_e(n)) and sent$_b$(v_e(n))={release} and s_{eb}(n)=release,

- winners(t_b(n))={e}=winners(image$_b$(v_e(n))) and queue(t_b(n)) and queue(image$_b$(v_e(n))) are empty and updated$_b$(v_e(n)) whenever $b \notin Fe \cap eF$ and not updated$_b$(v_e(n)) whenever $b \in Fe \cap eF$ and sent$_b$(v_e(n))=$\emptyset$ and either r_{eb}(n) is defined or s_{eb}(n)=none.

The following lemma can easily be verified.

Lemma 1. The subset C_0^* of markings $n \in C^*$ satisfying (i1) - (i10) is invariant under all transitions of N^*, i.e., $nf \in C_0^*$ for all $n \in C_0^*$ and $f \in F^*$.

Observe also that (i9) implies the following mutual exclusion property.

Lemma 2. For every $n \in C_0^*$, if two distinct $e, e' \in E$ are such that $(Fe \cup eF) \cap (Fe' \cup e'F) = \emptyset$ then $\underline{phase}(v_e(n)) = \underline{accessing}$ implies $\underline{phase}(v_{e'}(n)) \neq \underline{accessing}$.

Now, taking into account lemma 1, we define D^*, U, and W, as follows.

The subset D^* is defined as C_0^*.

For every $n \in D^*$ we define $U(n)$ in the following manner:

$$
(U(n))(b) = \begin{cases}
\underline{marking}(\underline{image}_b(v_e(n))) + 1 & \text{if } b \in Fe - eF \text{ and} \\
& \underline{phase}(v_e(n)) = \underline{accessing} \\
& \text{and } \underline{updated}_b(v_e(n)) \\
\underline{marking}(\underline{image}_b(v_e(n))) - 1 & \text{if } b \in eF - Fe \text{ and} \\
& \underline{phase}(v_e(n)) = \underline{accessing} \\
& \text{and } \underline{updated}_b(v_e(n)) \\
\underline{marking}(\underline{image}_b(v_e(n))) & \text{if } b \in Fe \cup eF \text{ and} \\
& \underline{phase}(v_e(n)) = \underline{accessing} \\
& \text{and not } \underline{updated}_b(v_e(n)) \\
\underline{marking}(t_b(n)) & \text{otherwise.}
\end{cases}
$$

Finally, given a string y of transitions of Q^* that is a real execution of Q^* starting from some $n \in D^*$, or an initial segment of such execution, we find the shortest initial segment z_1 of y such that:

$$z_1 = y_1 \underline{read}_{e_1} (\underline{line}_{b_1 e_1})$$

with $\underline{phase}(v_{e_1}(ny_1)) = \underline{accessing}$ and $\underline{fired}_{e_1}(v_{e_1}(ny_1))$ and $\underline{phase}(v_{e_1}(nz_1)) = \underline{releasing}$,

then the shortest initial segment z_2 of y such that:

$$z_2 = z_1 y_2 \underline{read}_{e_2} (\underline{line}_{b_2 e_2})$$

with $\underline{phase}(v_{e_2}(nz_1 y_2)) = \underline{accessing}$ and $\underline{fired}_{e_2}(v_{e_2}(nz_1 y_2))$ and $\underline{phase}(v_{e_2}(nz_2)) = \underline{releasing}$,

etc.

Thus we obtain a representation:

$$y = y_1 \underline{read}_{e_1} (\underline{line}_{b_1 e_1}) y_2 \underline{read}_{e_2} (\underline{line}_{b_2 e_2}) \ldots$$

and define $W(y)$ as the string $e_1 e_2 \ldots$. In the case when there is not any initial segment of y with required properties we define $W(y)$ as the empty string Λ.

It is obvious that D^*, U, and W, enjoy the properties (1) - (5). It remains to prove that they enjoy also the properties (6) - (8).

8. Correctness proof

We start with several properties of formal and real executions of the interpreted net Q^{*} of the system of modules.

Lemma 3. All formal executions of Q^{*} starting from $n \in D^{*}$ are infinite.

Proof. Let $y \in$ executions(Q^{*}, n) be finite. Then ny must be a dead marking. On the other hand, $ny \in D^{*}$ since D^{*} is invariant under all transitions of Q^{*}, and no marking belonging to D^{*} is dead. Q.E.D.

Lemma 4. Each read and each write transition occurs infinitely many times in each real execution of Q^{*} starting from $n \in D^{*}$.

Proof. Let $n \in D^{*}$ and $y \in$ realexecutions(Q^{*}, n). Consider a read or write transition, for example the transition write$_e$(line$_{eb}$). By the definition of D^{*}, either loaded(line$_{eb}$) carries a token $s_{eb}(n)$ or empty(line$_{eb}$) carries a token $r_{eb}(n)$. Let, for example, empty(line$_{eb}$) carries a token $r_{eb}(n)$. On the other hand, again by the definition of D^{*}, exactly one of at$_e$(line$_{eb'}$), at$_e$(line$_{b'e}$) ($b' \in Fe \cup eF$) carries a token $v_e(n)$. If at$_e$(line$_{eb}$) carries a token then write$_e$(line$_{eb}$) is enabled and, by axiom 2, it occurs in y, say as the last transition of an initial segment y_1 of y. If at$_e$(line$_{eb'}$) with $b' \neq b$ carries a token then skip$_e$(line$_{eb'}$) and possibly write$_e$(line$_{eb'}$) are enabled and, by axiom 1, at least one of these transitions must occur in y, say as the last transition of an initial segment z_1 of y. Similarly, if at$_e$(line$_{b'e}$) carries a token then skip$_e$(line$_{b'e}$) and possibly read$_e$(line$_{b'e}$) are enabled and at least one of these transitions must occur as the last transition of an initial segment z_1 of y. Since D^* is invariant, we obtain $nz_1 \in D^*$ and $y = z_1 y'$ with $y' \in \underline{\mathrm{realexecutions}}(Q^*.nz_1)$ and $\underline{\mathrm{at}}_e(\underline{\mathrm{next}}_e(\underline{\mathrm{line}}_{eb'}))$, or respectively $\underline{\mathrm{at}}_e(\underline{\mathrm{next}}_e(\underline{\mathrm{line}}_{b'e}))$, carrying a token $v_e(nz_1)$, and we can repeat the same reasoning. In a finite number of steps we come to a marking such that $\underline{\mathrm{at}}_e(\underline{\mathrm{line}}_{eb})$ carries a token, and we deduce that $\underline{\mathrm{write}}_e(\underline{\mathrm{line}}_{eb})$ occurs as the last transition of an initial segment y_1 of y.

In the case when $\underline{\mathrm{loaded}}(\underline{\mathrm{line}}_{eb})$ carries a token we take into account the fact that exactly one of $\underline{\mathrm{at}}_b(\underline{\mathrm{line}}_{e'b})$, $\underline{\mathrm{at}}_b(\underline{\mathrm{line}}_{be'})$ $(e' \in Fb \cup bF)$ carries a token $t_b(n)$, and deduce as before that $\underline{\mathrm{read}}_b(\underline{\mathrm{line}}_{eb})$ must occur in y, which leads us to the previous case.

Since D^* is invariant, we obtain $ny_1 \in D^*$ and a decomposition $y = y_1 y'$ with $y' \in \underline{\mathrm{realexecutions}}(Q^*.ny_1)$ and $\underline{\mathrm{write}}_e(\underline{\mathrm{line}}_{eb})$ occurring in y'. Thus we obtain infinitely many occurrences of $\underline{\mathrm{write}}_e(\underline{\mathrm{line}}_{eb})$ in y.

That other read and write transitions occur in y infinitely many times can be proved in similar way. Q.E.D.

Lemma 5. Two successive occurrences of a read transition $\underline{\mathrm{read}}_e(\underline{\mathrm{line}}_{be})$ or $\underline{\mathrm{read}}_b(\underline{\mathrm{line}}_{eb})$ (resp.: of a write transition $\underline{\mathrm{write}}_b(\underline{\mathrm{line}}_{be})$ or $\underline{\mathrm{write}}_e(\underline{\mathrm{line}}_{eb})$) in a real execution y of Q^* starting from $n \in D^*$ are separated by an occurrence of the corresponding write transition $\underline{\mathrm{write}}_b(\underline{\mathrm{line}}_{be})$ or $\underline{\mathrm{write}}_e(\underline{\mathrm{line}}_{eb})$ (resp.: of the corresponding read transition $\underline{\mathrm{read}}_e(\underline{\mathrm{line}}_{be})$ or $\underline{\mathrm{read}}_b(\underline{\mathrm{line}}_{eb})$).

Proof. The lemma follows from the fact that each read (resp.: write) transition is enabled only if the corresponding line is loaded (resp.: empty) and that such transition empties (resp.: loads) the line. Q.E.D.

Lemma 6. Given a marking $n \in D^*$ and a string z of read and write transitions such that:

(j1) each read and each write transition occurs in z infinitely many times,

(j2) two successive occurrences of a read (resp.: write) transition in z are separated by an occurrence of the corresponding write (resp.: read) transition,

(j3) if a communication line is empty (resp.: loaded) under n then each occurrence in z of the corresponding read (resp.: write) transition is preceded by an occurrence of the corresponding write (resp.: read) transition,

there exists a real execution y of Q^* starting from n such that z can be obtained by removing from y all skip transitions.

Proof. A real execution y as required can be defined by considering the successive initial segments of z and constructing inductively the corresponding initial segments of y.

Let $z=f_1 \ldots f_i f_{i+1} z$ and let y_i be a firing sequence of Q^* starting from n such that $f_1 \ldots f_i$ can be obtained by removing from y_i all skip transitions. Then $ny_i \in D^*$ and, by (j2) and (j3), either f_{i+1} is a read transition and the corresponding communication line is loaded or f_{i+1} is a write transition and the corresponding communication line is empty. In both cases there exists a firing sequence w starting from ny_i of skip transitions such that f_{i+1} is enabled under $ny_i w$. Defining y_{i+1} as $y_i w f_{i+1}$ we obtain a firing sequence of Q^* starting from n such that $f_1 \ldots f_i f_{i+1}$ can be obtained by removing from y_{i+1} all skip transitions.

Proceeding in this way we obtain a real execution y which enjoys the required property. Q.E.D.

Now we are ready to prove the properties (6) - (8) of D^*, U, and W, as defined in the previous section.

<u>Lemma 7</u>. $U(n)W(y)=U(ny)$ whenever $n \in D^*$ and ny is defined (property (6)).

Proof. If y is empty then W(y) is empty as well and $ny=n \in D^*$. Assuming that the property holds true for strings of the length $\leq i$ we consider y of the length i and prove that the property remains true if y is extended by one transition.

By the definition of W, the only non-trivial case is that of $y\underline{read}_e(\underline{line}_{be})$ with $\underline{phase}(v_e(ny))=\underline{accessing}$ and $\underline{fired}_e(v_e(ny))$ and $\underline{phase}(v_e(ny\underline{read}_e(\underline{line}_{be})))=\underline{releasing}$. Then $W(y\underline{read}_e(\underline{line}_{be}))=W(y)e$ and, by (19) and the definition of U, ny is such that e is enabled under $U(ny)=U(n)W(y)$ and $U(ny\underline{read}_e(\underline{line}_{be}))=U(ny)e=U(n)W(y)e=U(n)W(y\underline{read}_e(\underline{line}_{be}))$. Q.E.D.

Lemma 8. Given a string x of transitions of P that is a finite execution of P starting from a marking m, or a finite initial segment of an infinite execution starting from m, there exist $n \in D^*$ and a string y of transitions of Q^* such that U(n)=m, y is a real execution of Q^* starting from n, or respectively an initial segment of such execution, and x=W(y) (property (8)).

Proof. We choose $n \in D^*$ such that the following conditions are fulfilled for all $b \in B$ and $e \in Fb \cup bF$:

- $\underline{phase}(v_e(n))=\underline{waiting}$.

- $\underline{queue}(\underline{image}_b(v_e(n)))=\underline{queue}(t_b(n))$ are empty,

- $\underline{winners}(\underline{image}_b(v_e(n)))=\underline{winners}(t_b(n))=\emptyset$,

- $\underline{marking}(\underline{image}_b(v_e(n)))=\underline{marking}(t_b(n))=m(b)$,

- not $\underline{updated}_b(v_e(n))$.

- $\underline{sent}_b(v_e(n))=\emptyset$ and $r_{eb}(n)$ is defined.

Then U(n)=m.

If x is an empty execution of P then m must be a dead marking and, by the definition of W, W(y) is empty for every real execution y of Q^* starting from n. Since such real executions exist by lemma 6, we obtain x=W(y) for any of them.

If x is a finite initial segment of a non-empty execution of P then we proceed as follows.

If x is empty then we choose the empty string for y and we obtain x=W(y).

Let x be non-empty.

Consider the initial segment $e_1 \dots e_{i-1} e_i$ of x and suppose that some y as required has been found for the initial segment $e_1 \dots e_{i-1}$ such that ny fulfils the conditions imposed on n. Then $\underline{\text{enabled}}_{e_i}(v_{e_i}(ny))$ and, by lemma 6, a string w of transitions of Q^{*} can be found such that:

- yw is an initial segment of a real execution of Q^{*} starting from n,
- the transitions of w carry $\underline{\text{control}}_{e_i}$ (and only this module) over the phases <u>waiting</u>, <u>distributing</u>, <u>winning</u>, <u>cleaning</u>, <u>accessing</u>, and <u>releasing</u>,
- nyw fulfils the conditions imposed on n,
- $W(yw)=W(y)e_i$.

Thus some y as required can be found for all initial segments of x and for x itself. Moreover, if x is a finite execution of P then mx is a dead marking and the obtained string y can be extended to a real execution of Q^{*} without any change of W(y). Q.E.D.

<u>Lemma 9</u>. If $n \in D^{*}$ and $y \in \underline{\text{realexecutions}}(Q^{*},n)$ then $W(y) \in \underline{\text{executions}}(P,U(n))$ (property (7)).

Proof. By lemma 3, y is an infinite string $f_1 f_2 \dots$. We have to prove that if a transition of P is enabled under $U(nf_1 \dots f_i)$ then $U(nf_1 \dots f_i)e=U(nf_1 \dots f_i \dots f_{i+k})$ for some $k>0$ and $e \in E$.

By lemma 4, each read and write transition occurs in y infinitely many times. Together with the invariance of D^{*} and the properties (i1) - (i10) of markings belonging to D^* it implies that all signals sent by modules controlling transitions of P to modules controlling adjacent places lead to the expected effects in the corresponding places, and that these effects are discovered by the senders. In particular, each module assigned to a transition acts according to the scheme shown in fig. 7.

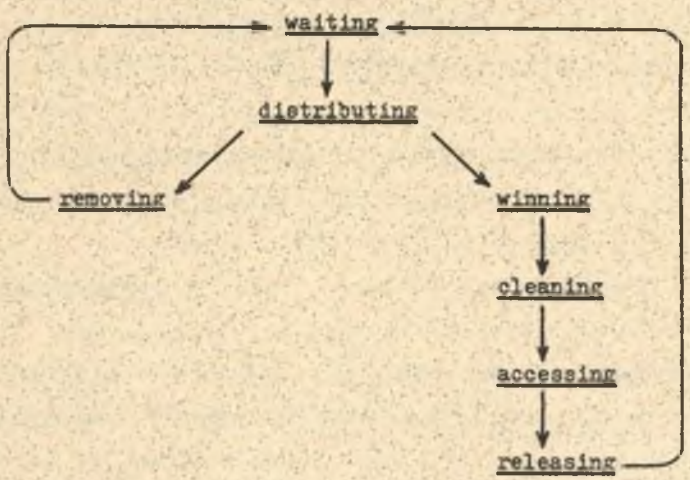

Fig. 7

In order to prove that $U(nf_1\ldots f_i)e = U(nf_1\ldots f_i\ldots f_{i+k})$ for some $k>0$ and $e\in E$ suppose the contrary. Then, for all $k>0$, we have $U(nf_1\ldots f_i\ldots f_{i+k}) = U(nf_1\ldots f_i)$ and there are two possibilities:

- all the modules controlling transitions are blocked in the sense that they cannot move (change situations in adjacent places or move from one phase to another),
- there are modules controlling transitions that are not blocked but all of them act only in phases <u>waiting</u>, <u>distributing</u>, and <u>removing</u>.

In the first case a marking $n' \in F^*$ is reached such that all the modules controlling transitions are blocked. Then, by (i1) - (i10), it must be $\underline{image}_b(v_e(n'))=t_b(n')$ for all $e \in E$ and $b \in Fe \cup eF$.

Now, by (i1) - (i10), there is not any $e \in E$ whose control is in one of the phases winning, cleaning, accessing, releasing (otherwise the corresponding module could move). In particular, we have $\underline{phase}(v_e(n')) \in \{\underline{waiting}, \underline{distributing}, \underline{removing}\}$ and $\underline{winners}(\underline{image}_b(v_e(n')))=\underline{winners}(t_b(n'))=\emptyset$ for all $e \in E$ and $b \in Fe \cup eF$.

Again by (i1) - (i10), there is not any $e \in E$ whose control is in the phase removing. So $\underline{phase}(v_e(n'))=\underline{distributing}$ for a non-empty subset E' of enabled transitions of F and $\underline{phase}(v_e(n'))=\underline{waiting}$ for all $e \in E-E'$. This implies that all $e \in E' \cap (Fb \cup bF)$, and only such transitions, occur in each $\underline{queue}(t_b(n'))$ and that the transitions of higher priorities follow those of lower priorities (otherwise there would be a module which could move).

From the assumption that the modules controlling transitions are blocked it follows that, for each $e \in E'$ there exist $b \in Fe \cup eF$ and $e' \in E'$ such that e' precedes e in $\underline{queue}(t_b(n'))$. Thus we obtain an infinite sequence $e_1, e_2, \ldots$ of transitions from E' such that for each e_i there exist $b_i \in Fe_i \cup e_iF$ and e_{i+1}, where e_{i+1} precedes e_i in $\underline{queue}(t_{b_i}(n'))$. On the other hand, we have seen that in such a case the priority of e_{i+1} must be lower than that of e_i. So each transition occurs in the infinite sequence $e_1, e_2, \ldots$ at most once, which is impossible for finite E.

In the second case the modules controlling enabled transitions of F would only distribute and remove their visiting-cards in adjacent places (by sending signals card and cardback, respectively). However, this cannot continue infinitely long since in such a case the control of highest priority would never be obliged to remove its visiting-cards, and so it would become a winner and access adjacent places in spite of our assumption. Q.E.D.

All the results we have obtained can be summarized as follows.

Theorem. The implementation of a Petri net P by a system of modules as described in sections 2 - 6 is correct in the sense that there are D^{*}, U, and W, which enjoy the properties (1) - (8) of section 7.

9. Final remarks

In order to implement a Petri net P we have constructed a system of modules controlling places and transitions of P. The behaviour of the system of modules has been described with the aid of the corresponding interpreted net Q^{*} and its executions.
In order to reflect the necessary physical properties of the system of modules we have restricted ourselves to the real executions which satisfy suitable axioms. Such real executions are all infinite and each of them contains infinitely many occurrences of each read and write transition so that no module is dead. To each real execution of Q^{*} there corresponds a movement of tokens in Q^{*} and a process of transforming the information contained in such tokens. A part of this information represents a marking of the implemented net P and is being changed as if P would be executed. This process continues while enabled transitions of P can be found. It continues also if a dead marking of P is reached but then it goes in vain (the part of information which represents a marking of P does not change anymore).

The reletionships between the executions of P and the real executions of Q^{*} have been expressed with the aid of an invariant subset D^{*} of markings of Q^{*} and two functions U and W describing how markings and firing sequences of P are represented by those of Q^{*}. We have expressed these relationships by the properties (1) - (8) in section 7, taking into account the fact that independent transitions can be executed in parallel even though they occur one after another in the string representing the considered execution.

Each execution of a transition of P is represented by a subsequence of the corresponding real execution of Q^{*}. Such subsequence consists of the transitions the executing module performs between winning the access to adjacent places and leaving the phase of accessing (cf. the definition of W in section 7, where the executed transition of P is represented by the last transition of the corresponding subsequence). The subsequences representing executions of independent transitions may overlap as shown in fig. 6, which reflects the parallelism of executions. Due to such overlapping the respective parts of the successive marking of P may hold valid in different or even disjoint time intervals (cf. the remark about markings in section 1). Each successive marking of Q^{*}, whose parts may also hold valid in different time intervals, describes which transitions of P are in progress and how far is the progress (cf. the definition of U, where the information contained in a marking $n \in D^{*}$ of Q^{*} has been used to reconstruct the last marking of each place of P participating in a transition).

$$f_1f_2\ldots\overbrace{f_i\ldots\mathrlap{\underbrace{\phantom{f_j\ldots f_k=\underline{\text{read}}_{e_1}(\underline{\text{line}}_{b,e_1})\ldots f_l=\underline{\text{read}}_{e_2}(\underline{\text{line}}_{b_2e_2})}}_{\text{an execution of } e_2}}f_j\ldots f_k=\underline{\text{read}}_{e_1}(\underline{\text{line}}_{b,e_1})}^{\text{an execution of } e_1}\ldots f_l=\underline{\text{read}}_{e_2}(\underline{\text{line}}_{b_2e_2})\ldots$$

Fig. 8

Observe that in the subsequences representing parallel executions of independent transitions e_1 and e_2 of P there may be occurrences of transitions f_p and f_q of Q^*, respectively, which are causally dependent. This is an unintended indirect effect of the useful communication among the modules. Unfortunately, such indirect effect has also some undesirable consequences. We shall illustrate them on the example of marked net in fig. 9.

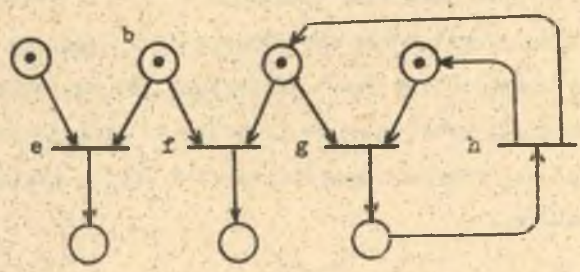

Fig. 9

Suppose that the control of f is of higher priority than that of e and that the controls of g and h are much faster than those of e and f. Then it is likely that the controls of g and h win the access to adjacent places always whenever enabled so that the control of f never wins. On the other hand, it may happen that after some executions of g and h the control of f deposits its visiting-card in place b and that the visiting-card of the control of e is not present there yet. Then the control of e must remove the cards it has possibly distributed and return to waiting. Next the control of f loses and removes its visiting-card from b, which allows the control of e to start distributing cards again, and a similar sequence of events may follow infinitely many times. Thus we have the permanently enabled transition e which never is executed.

Such phenomenon is not excluded by the definition of execution in section 1. Nevertheless, we feel that a stronger concept of execution of a net is necessary for certain purposes. For instance, the correctness of our implementation has been proved with the aid of axiom 1 in section 7 whose role was just to exclude permanently enabled but never executed transitions. So, in order to be consistent, we should rather use the concept of a <u>complete</u> execution, where the completeness means satisfying the mentioned axiom. With such concept it would also be possible to strengthen lemma 8 and prove that all complete executions of P can be realized by the system of modules.

The example in fig. 9 shows that our implementation does not guarantee the completeness of executions. It is also an open question how to modify the implementation in order to remove this insufficiency.

In the paper we have restricted the problem of implementation to uninterpreted Petri nets. However, after slight modifications allowing to reconstruct not only the last markings of places of P participating in transitions but also the information contained is such markings, our solution applies to interpreted nets as well. Thus we obtain a general method of implementing systems of activities of a broad class. Regarding such systems as programs we could develop a programming language and tell how to implement it in order to get programs executed efficiently.

The result we have obtained illustrates an interesting possibility. It shows that complex problems of synchronization can be solved with the aid of very simple modules and very simple communication.

References

1. P. Furtek: Modular implementation of Petri nets, MS thesis, Dept. Electrical Engineering, MIT, Cambridge, Mass., 1971

2. H.J. Genrich, K. Lautenbach, and P.S. Thiagarajan: Elements of general net theory, Lecture Notes in Comp. Science 84 (Springer, Berlin, 1980)

3. A.W. Holt et al.: Final report on the Project of Information System Theory, Appl. Data Research ADR 606 and USAF - Rome Air Development Centre, RADC-TR-68-305, 1968

4. A. Mazurkiewicz: Concurrent program schemes and their interpretations, DAIMI PB-78, Aarhus University, 1977

5. C.A. Petri: Non-sequential processes, Interner Bericht ISF-77-5, Gesellschaft fuer Math. und Datenverarbeitung, 5205 St. Augustin, W. Germany, 1977

6. C.A. Petri: Concurrency as basis of system thinking, Proc. 5th Scandinavian Logic Symp., Aalborg Universitetsforlag, 1979

7. L. Priese: Automatatheoretical approach to concurrency, Internal Report, Ser. B, No. 12 (Digital Systems Lab. of the Helsinki University of Technology, 1980

8. J. Winkowski: Behaviours of concurrent systems, Theoret. Comp. Sci. 12(1), 1980

9. J. Winkowski: An algebraic description of system behaviours, Theoret. Comp. Sci. 21, 1982

10. J. Winkowski: Protocols of accessing overlapping sets of resources, Inf. Processing Lett. 12(5), 1981