LIVENESS OF EXTENDED CONTROL STRUCTURE NETS

by

Otthein Herzog

USUU-77 ~~107~~
119

August 1977

ABSTRACT

A new subclass of Petri nets is defined allowing the control structure representation of parallel programs including arbitrary semaphore operations. Structural properties of the "Extended Control Structure Nets" are discussed and seven necessary and sufficient conditions for their liveness are proved.

This work was supported in part by the Deutsche Forschungsgemeinschaft Bonn through grant He/589/2.

LIVENESS OF EXTENDED CONTROL STRUCTURE NETS

Otthein Herzog
Abteilung Informatik
Universität Dortmund
Postfach 500500
D-4600 Dortmund 50
Fed. Rep. of Germany

1. INTRODUCTION

There exist various programming languages offering the feature of parallel programming like PL/I, BURROUGHS Extended Algol, ALGOL 68, Concurrent Pascal etc. While in the first two mentioned programming languages the synchronization is done by boolean event variables, variables of type "semaphore" /4/ are offered for that purpose in ALGOL 68 and the synchronization constructs of Concurrent Pascal /1/ can be implemented using semaphores.
In /8/ and /9/, a subclass of the Petri nets, the Control Structure Nets are studied suited to the modeling of parallel programs using only event variables for the synchronization between the different tasks.
In this paper, the class of Extended Control Structure Nets is defined allowing the control structure representation of parallel programs including semaphore operations on semaphore variables.

The main part of this work was done while the author was as research associate with the Computer Science Department of the University of Utah.

This new subclass is defined recursively by a composition of connected state machines. Some structural properties of Extended Control Structure Nets are shown and seven necessary and sufficient conditions for the liveness are proved.

2. GRAPHS OF EXTENDED CONTROL STRUCTURE NETS

2.1. Basic definitions

Definition 1:

The graph of a Petri net gpn = (P, T; PRE, POST) consists of

- the finite, nonempty set P ("places"),
- the finite set T ("transitions"),

  where $P \cap T = \emptyset$, and
- the functions PRE: $P \times T \to \mathbb{N} \cup \{0\}$

  and POST: $T \times P \to \mathbb{N} \cup \{0\}$.

According to this definition, a graph of a Petri net may consist of exactly one place.

Usually, in a graphical representation, places are drawn as circular, transitions as rectangular nodes. The value of PRE, resp. POST of a pair of places and transitions will be attached to the connecting arc.

The following definition fixes the graph-theoretical notations used throughout this paper.

Definition 2:

Let gpn = (P, T; PRE, POST) be the graph of a Petri net;

$x_i \in P \cup T$, $i = 1,\ldots,n$.

1. $(x_1,\ldots,x_n)$ is called sequence of edges ("se") iff

   (i) $x_1, x_n \in P$

   (ii) $PRE(x_i,x_{i+1}) > 0$ or $POST(x_i,x_{i+1}) > 0$ $(i=1,\ldots,n-1)$

2. A sequence of edges $se=(x_1,\ldots,x_n)$ is called path ("pa") iff

   $(\forall\, i\neq j \in \{1,\ldots,n\})\ (\forall (x_i,x_{i+1}),(x_j,x_{j+1})): (x_i,x_{i+1}) \neq (x_j,x_{j+1})$

3. A path $pa=(x_1,\ldots,x_n)$ is called simple path ("sp") iff

   $(\forall\, i\neq j \in \{1,\ldots,n\})\ (\forall x_i, x_j): x_i \neq x_j$

4. A path $pa=(x_1,\ldots,x_n)$ is called elementary circuit ("ec") iff

   (i) $(\forall\, i\neq j \in \{2,\ldots,n\})(\forall x_i, x_j): x_i \neq x_j$

   (ii) $x_1 = x_n$

5. For a sequence of edges $se=(x_1,\ldots,x_n)$,

   (i) $V(se) := \{ x_i \in P \cup T \mid i \in \{1,\ldots,n\} \}$ ("set of vertices of se")

   (ii) $E(se) := \{ (x_i,x_{i+1}) \mid i \in \{1,\ldots,n-1\} \}$ ("set of edges of se")

6. $E(gpn) := \{ (x_i,x_{i+1}) \mid PRE(x_i,x_{i+1}) \neq 0 \text{ or } POST(x_i,x_{i+1}) \neq 0 \}$

   ("set of edges of gpn")

Now, the graph of a connected Petri net is defined. An example is given afterwards.

Definition 3:

Let gpn = (P, T; PRE, POST) be a graph of a Petri net;

let $p_I \in P$ be an initial place, where $(\forall t \in T)$: $POST(t,p_I) = 0$,

$p_F \in P$ be a final place, where $(\forall t \in T)$: $PRE(p_F,t) = 0$,

P_I be the set of initial places, P_F be the set of final places.

1. gpn is called graph of a connected Petri net ("gcpn") iff

   (i) $|P_I| = 1, |P_F| \geq 1$

   (ii) $(\forall p \in P)$ $(\exists$ simple path $sp_I)$: $sp_I = (p_I,\ldots,p)$

   (iii) $(\forall p \in P)$ $(\exists$ simple path $sp_F)$: $sp_F = (p,\ldots,p_F)$

2. $GCPN := \{ gpn \mid gpn$ is graph of a connected Petri net $\}$

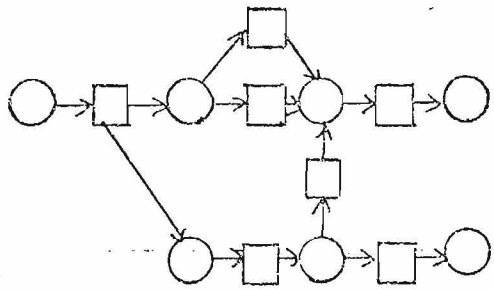


Fig.1: Instance of a graph of a connected Petri net

Further notations:

Let gcpn $\in$ GCPN, gcpn = (P,T; PRE,POST); $p \in P$, $t \in T$.

1. $PRED(t) := \{ p \in P \mid PRE(p,t) > 0 \}$

   $SUCC(t) := \{ p \in P \mid POST(t,p) > 0 \}$

   $PRED(p) := \{ t \in T \mid POST(t,p) > 0 \}$

   $SUCC(p) := \{ t \in T \mid PRE(p,t) > 0 \}$

2.(i) t is called signal transition ("s-transition") iff $|SUCC(t)| > 1$

(ii) $T \supseteq S := \{ t \mid t \text{ is s-transition} \}$

3.(i) t is called receive transition ("r-transition") iff $|PRED(t)| > 1$

(ii) $T \supseteq R := \{ t \mid t \text{ is r-transition} \}$

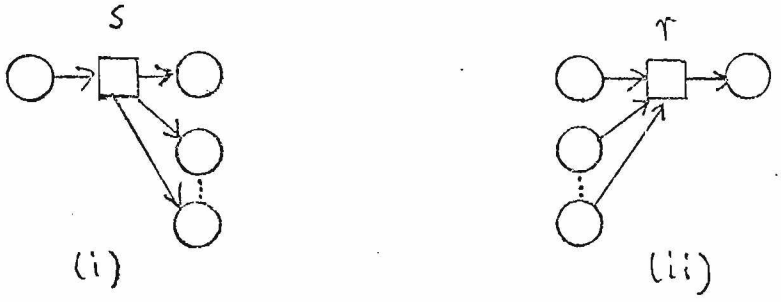


Fig.2: (i) s-transition s, (ii) r-transition r

By the next definition graphs of connected state machines (/6/ and /10/) are defined which are very closely related to state graphs of finite automata.

Definition 4:

Let gcpn GCPN, gcpn = (P,T; PRE,POST).

1. gcpn is called graph of a connected state machine ("gcsm") iff

   (i) $(\forall\, p \in P)\ (\forall\, t \in T)$: $PRE(p,t) \leq 1$

   (ii) $(\forall\, t \in T)\ (\forall\, p \in P)$: $POST(t,p) \leq 1$

   (iii) $(\forall\, t \in T)$: $|PRED(t)| = |SUCC(t)| = 1$

   (iv) $|P_F| = 1$

2. $GCSM := \{ gcpn \in GCPN \mid gcpn \text{ is a graph of a connected state machine} \}$

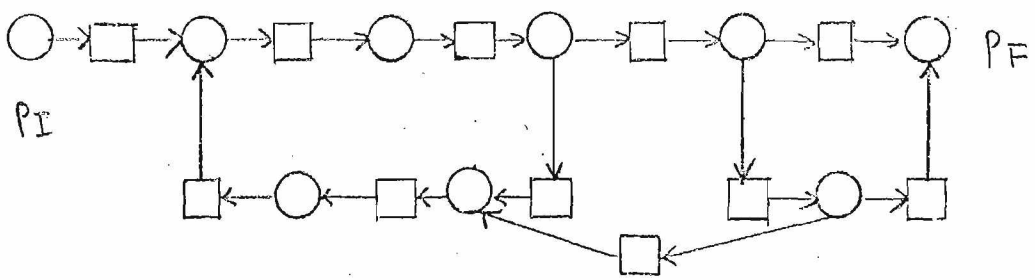


Fig.3: Instance of a graph of a connected state machine

Such a connected state machine can be thought as representing a purely sequential part of a program like a sequential procedure. It is easy to see that arbitrary sequential control structures can be modeled far away from any structured programming.

Definition 5 explaines the composition of two graphs of connected Petri nets:

- the sets of places and transitions have to be disjoint;
- the already existing arcs are preserved;
- if both graphs of connected Petri nets contain transitions then exactly one arc has to be added outgoing from an arbitrary transition of the first graph and entering the initial place of the second graph;
- if the second graph consists of only one place then arbitrary many arcs can be added
  - outgoing from the first graph and entering the only place of the second one,
  - as well as outgoing from the place of the second graph entering arbitrary transitions of the first one.

This composition seems to be quite restrictive, but after a look to the underlying motivation it appears to be rather natural:
the composition is intended to give the possibility of modeling the attaching of a subtask to a calling task. This requires

- disjointness of the code of these two tasks (this does not prevent the multiple attaching of the same task as subtasks: if the code is written reentrantly the representation of that code can be copied upon each request.
- representation of the attaching mechanism, namely starting the subtask at exactly the initial statement.
- synchronization in many cases which can be done by variables, i.e. event variables or semaphores represented by the single place.

Examples of the composition follow definition 5 (fig.4).

Definition 5:

Let gpn = (P,T; PRE,POST) be a graph of a Petri net,

$gcpn_i \in GCPN$, $gcpn_i = (P^i, T^i; PRE^i, POST^i)$, $(i=1,2)$, where $P^1 \cap P^2 = T^1 \cap T^2 = \emptyset$;

P^i_I the set of initial places of $gcpn_i$;

$U^1 \subseteq T^1$: $|U^1| = 1$.

$gpn = gcpn_1 + gcpn_2$ is called <u>composition of $gcpn_1$ and $gcpn_2$</u> iff

(i) $P = P^1 \cup P^2$

(ii) $T = T^1 \cup T^2$

(iii)
$$PRE := \begin{cases} (P^1 \cup P^2) \times (T^1 \cup T^2) \longrightarrow \mathbb{N} \cup \{0\} \\ (p,t) \longrightarrow \begin{cases} PRE^i(p,t), \text{ if } p \in P^i,\ t \in T^i \\ n \in \mathbb{N} \cup \{0\}, \text{ if } \{p\} = P^2,\ t \in T^1 \\ 0 \text{ else} \end{cases} \end{cases}$$

(iv)
$$POST := \begin{cases} (T^1 \cup T^2) \times (P^1 \cup P^2) \longrightarrow \mathbb{N} \cup \{0\} \\ (t,p) \longrightarrow \begin{cases} POST^i(t,p), \text{ if } t \in T^i,\ p \in P^i \\ 1, \text{ if } \{t\} = U^1,\ \{p\} = P^2_I \subset P^2 \\ n \in \mathbb{N} \text{ if } \{t\} = U^1,\ \{p\} = P^2 \\ n \in \mathbb{N} \cup \{0\} \text{ if } t \in T^1,\ \{p\} = P^2 \\ 0 \text{ else} \end{cases} \end{cases}$$

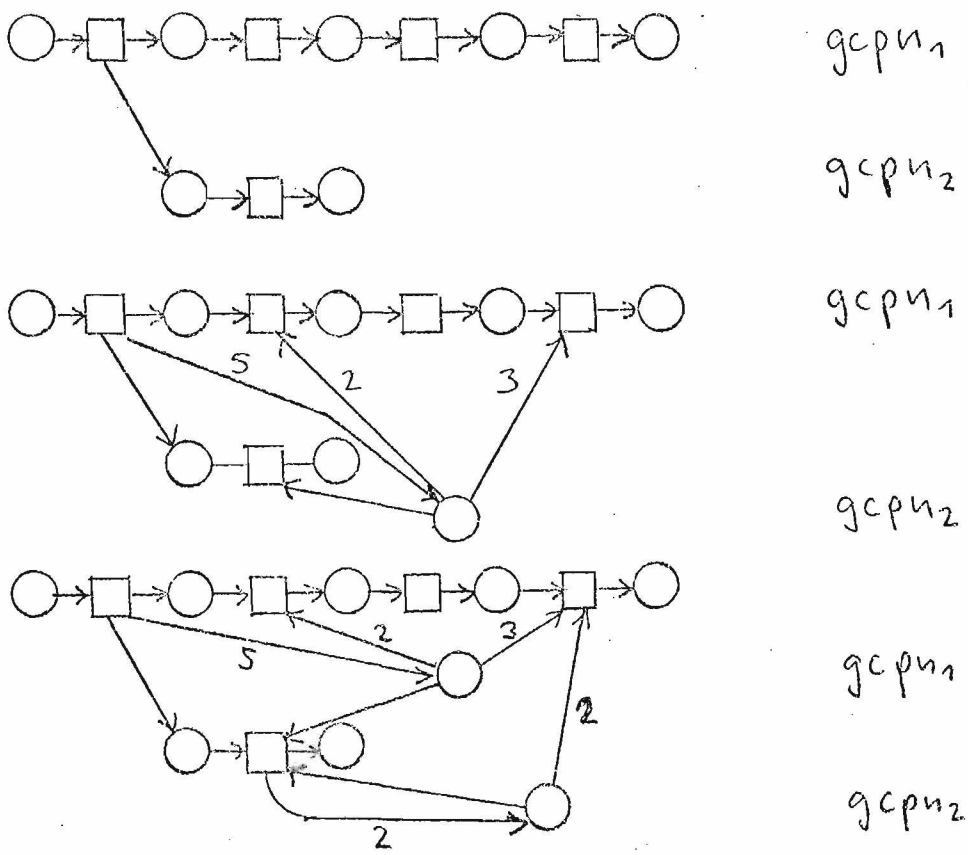


Fig.4: Examples for the composition

Lemma 1 shows that by a composition there can be arcs added only, and that at least one arc has to be added at a composition.

Lemma 1:

Let $gcpn_i \in GCPN$, $gcpn_i = (P^i, T^i; PRE^i, POST^i)$ $(i = 1,2)$;

$gpn = (P,T; PRE,POST)$ a graph of a Petri net: $gpn = gcpn_1 + gcpn_2$.

Then

1. $E(gcpn_1) \cup E(gcpn_2) \subset E(gpn)$
2. $|E(gpn)| \geq |E(gcpn_1)| + |E(gcpn_2)| + 1$

Proof:

1. By def.5 (iii) and (iv),

$$E(gpn) \supseteq E(gcpn_1) \cup E(gcpn_2) \cup \left\{(u,p_I^2) \mid \{u\} = U^1, \{p_I^2\} = P_I^2\right.$$

2. follows directly from 1.

In lemma 2 it is stated that the resulting graph of the composition of two connected graphs of a Petri net is a graph of a connected Petri net, too.

Lemma 2:

Let $gcpn_i \in GCPN$, $gcpn_i = (P^i,T^i;\ PRE^i,POST^i)$ $(i=1,2)$;

$gpn = (P,T;\ PRE,POST)$ a graph of a Petri net: $gpn = gcpn_1 + gcpn_2$.

Then

$gpn \in GCPN$

Proof:

It has to be shown that gpn satisfies the conditions of def. 3:

(i) Let $\{p_I^i\} = P_I^i \subseteq P^i$ $(i=1,2)$.

By def.5, $POST(u,p_I^2) = 1$ where $\{u\} = U^1$,

$POST(t,p_I^2) = 0$ where $t \in T$.

Thus, $P_I = P_I^1$

(ii) Let $p_F^1 \quad P_F^1$.

By def.5, $(\forall\, t \in T) : PRE(p_F^1,t) = 0$,

by def.3, $|P_F^1| \geq 1$.

Thus, $|P_F| \geq 1$

(iii) Let $u = U^1$, $p \in P^1$: $p \in PRED(u)$.

By def.3, $sp_I = (p_I^1,\ldots,p)$ exists and

$(\forall p' \in P^2)$(simple path sp_I'): $sp_I' = (p_I^2,\ldots,p')$.

Thus, $(\forall p' \in P^2)(\exists$ simple path $sp'')$: $sp'' = (p_I^1,\ldots,p,u,p_I^2,\ldots,p')$

(iv) 1. By def.5, $(\forall t' \in T^2)$: $PRE(p_F^1,t') = 0$

Thus, $(\forall p \in P^1)$ $(\exists\ p_F^1 \in P_F^1)$ $(\exists$ simple path $sp_F)$: $sp_F = (p,\ldots,p_F^1)$

2.1. $|P^2| > 1$:

By def.5, $(\forall t \in T^1)$: $PRE(p_F^2,t) = 0$

Thus, $(\forall p' \in P^2)$ $(\exists\ p_F^2 \in P_F^2)$ $(\exists$ simple path $sp_F')$: $sp_F' = (p',\ldots,p_F^2)$

2.2. $|P^2| = 1$:

if $(\exists\ t \in T^1)$: $PRE(p^2,t) \neq 0$

then $(\exists p \in SUCC(t): p \in P^1)$ $(\exists\ p_F^1 \in P_F^1)$ $(\exists$ simple path $sp_F'')$: $sp_F'' = (p^2,t,p,\ldots,p$

Now, graphs of Extended Control Structure Nets are defined recursively by the composition of graphs of connected state machines.

Definition 6:

1. (i) gcsm GCSM is called graph of an Extended Control Structure Net ("gecsn")

(ii) gecsn + gcsm, gcsm GCSM is called graph of an Extended Control Structure Net.

(iii) Any graph of a Petri net obtained by a finite number of compositions of a graph of an Extended Control Structure Net and a graph of a connected state machine is called graph of an Extended Control Structure Net.

2. GECSN := $\{$ gpn graph of a Petri net $|$ gpn is a gecsn $\}$

2.2. Structural properties of Extended Control Structure Nets

First it is stated that a graph of an Extended Control Structure Nets belongs to the class of connected graphs of Petri nets.

Lemma 3:

Let $gecsn \in GECSN$, $gecsn = (\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i; PRE, POST)$.

Then

$gecsn \in GCPN$

Proof:

By induction:

(i) $gecsn \in GCSM$ by def.6, and by def.4, $gecsn \in GCPN$

(ii) Assume $gecsn_1 \in GECSN$, $gecsn_1 = \sum_{i=1}^{l-1} gcsm_i$, where $gcsm_i \in GCSM$ and $gecsn \in GCPN$

(iii) Let $gecsn_1 = (P^1, T^1; PRE^1, POST^1)$, $gcsm_2 \in GCSM$, $gcsm_2 = (P^2, T^2; PRE^2, POST^2)$.

$gecsn = gecsn_1 + gcsm_2$

By lemma 2, $gecsn \in GCPN$, as $gcsm_2 \in GCPN$

In the following two definitions, internal and external paths of graphs of Extended Control Structure Nets are defined, where an internal path contains only vertices belonging to exactly one component of the composition, i.e. to exactly one graph of a connected state machine, whereas an external path contains vertices belonging to at least two components.

Examples are given in fig.5 and fig.6.

Definition 7:

Let gecsn $\in$ GECSN, gecsn = $(\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i; PRE, POST)$;

$p,q \in \bigcup_{i=1}^{l} P^i$.

1. (i) A path pa = (p,...,q) is called internal path ("ipa") iff

$(\exists\, j \in \{1,\ldots,l\})\ (\forall\, x \in V(pa)): x \in P^j \cup T^j$

(ii) $IPA_j := \{ pa = (p,\ldots,q) \mid pa \text{ is internal path}, j \in \{1,\ldots,l\} \}$

2. (i) An internal path ipa = (p,...,q) is called simple internal path ("sipa") i

ipa is a simple path

(ii) $SIPA_j := \{ ipa \in IPA_j \mid ipa \text{ is simple path} \}$

3. (i) An internal path ipa = (p,...,q) is called loop ("lp") iff

ipa is an elementary circuit

(ii) $LP_j := \{ ipa \in IPA_j \mid ipa \text{ is elementary circuit} \}$

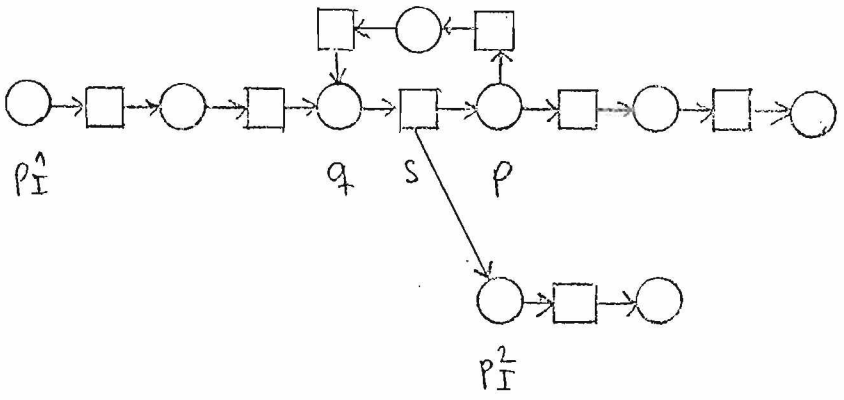


Fig.5: Internal paths

In fig.5, $(p_I^1,\ldots,q,s,p,t,\ldots,q) \in IPA_1$, $(p_I^1,\ldots,q,s,p) \in SIPA_1$,

$(q,s,p,\ldots,q) \in LP_1$ and $(p_I^1,\ldots,q,s,p_I^2) \notin IPA_1$.

Definition 8:

Let gecsn $\in$ GECSN, gecsn = $(\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i;$ PRE, POST);

$p,q \in \bigcup_{i=1}^{l} P^i$.

1.(i) A path pa = (p,...,q) is called external path ("epa") iff

$(\exists\, j \neq k \in \{1,\ldots,l\})\ (\exists\, x,y \in V(pa))$: $x \in P^j \cup T^j$, $y \in P^k \cup T^k$

(ii) $EPA_{j,k} := \{ pa=(p,\ldots,q) \mid$ pa is external path containing vertices from $P^j \cup T^j$ and $P^k \cup T^k \}$

2.(i) epa $\in EPA_{j,k}$ is called simple external path iff

epa is a simple path.

(ii) $SEPA_{j,k} := \{ epa \in EPA_{j,k} \mid$ epa is simple path $\}$

3.(i) epa $\in EPA_{j,k}$, epa = (p,...,q), $p \in P^j$ is called synchronization circuit ("sct

iff

(p,...,q) is an elementary circuit

(ii) $SCT_j := \{ epa \in EPA_{j,k} \mid$ epa is synchronization circuit and
$(\exists\, r \in R: r \notin V(epa), r \in T^m)\ (\exists\, q' \in PRED(r): q' \in P^m, q' \notin V(epa))$
$(\exists\, sipa \in SIPA_m$: sipa=$(p_I^m,\ldots,q'))$: $V(sipa) \cap V(epa) = \emptyset$ $\}$

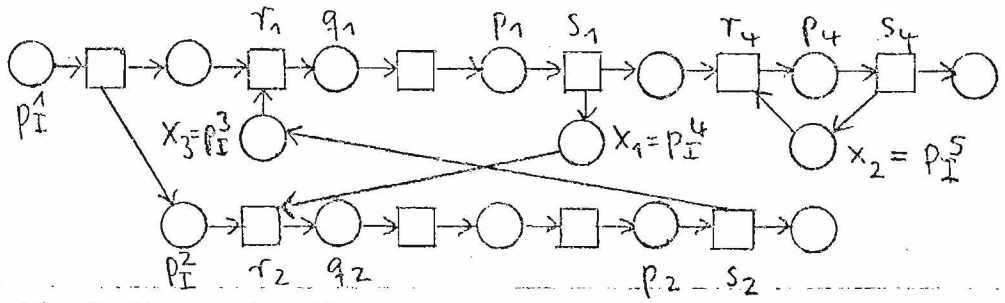


Fig.6: External paths

In fig.6, $(p_1,s_1,x_1) \in SEPA_{1,4}$, $(p_1,s_1,x_1,r_2,q_2,\ldots,p_2,s_2,x_3) \in SEPA_{1,3}$,
$(p_4,s_4,x_2) \in SEPA_{1,5}$, $(p_1,s_1,x_1,r_2,q_2,\ldots,p_2,s_2,x_3,r_1,q_1,\ldots,p_1) \in SCT_1$ and
$(p_4,s_4,x_2,r_4,p_4) \in SCT_1$

By definition 9 it is explained what is understood as non-alternating simple paths essentially being simple paths which cannot come back to a component once having left it. Examples are given in fig.7.

Definition 9:

Let gecsn $\in$ GECSN, gecsn = $(\dot{\bigcup}_{i=1}^{l} P^i, \dot{\bigcup}_{i=1}^{l} T^i; \text{PRE}, \text{POST})$;

$k \in \{1,\ldots,l\}$.

1. sp=(p,...,q) is called non-alternating simple path ("nasp") iff

   $(\forall\, t \in V(sp)\colon t \in T^k)\ (\forall\, p' \in \text{PRED}(t)\colon p' \in V(sp))\colon p' \in P^k$

2. $\text{NASP}_{p,q} := \{ sp = (p,\ldots,q) \mid$ sp non-alternating simple path from p to q $\}$

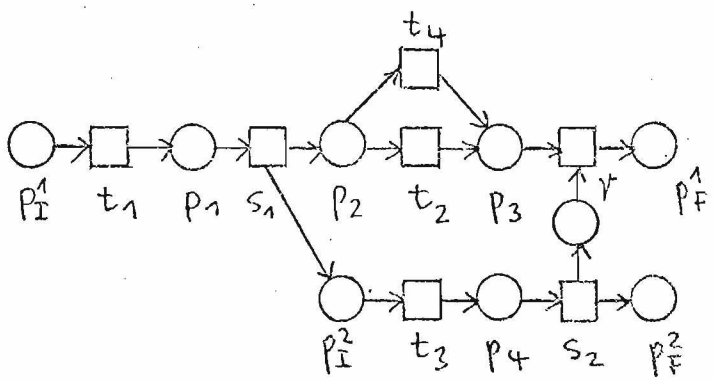


Fig.7: Non-alternating simple paths

In fig.7, $(p_I^1, t_1, p_1, s_1, p_2, t_2, p_3, r, p_F^1) \in \text{NASP}_{p_I^1, p_F^1}$,

$(p_I^1, t_1, p_1, s_1, p_2, t_4, p_3, r, p_F^1) \in \text{NASP}_{p_I^1, p_F^1}$,

$\{(p_I^1, t_1, p_1, s_1, p_I^2, t_3, p_4)\} = \text{NASP}_{p_I^1, p_4}$,

$(p_I^1, t_1, p_1, s_1, p_I^2, t_3, p_4, s_2, p_I^3, r, p_F^1) \notin \text{NASP}_{p_I^1, p_F^1}$ because of $\{p_I^3\} = P^3$, $r \in T^1$.

By the following lemma 4, some properties are given for the "environment" of s-transitions.

Lemma 4:

Let $gecsn \in GECSN$, $gecsn = (\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i ; PRE, POST)$;

$j \in \{1,\ldots,l\}$;

$s \in S: s \in T^j$, $q \in PRED(s): q \in P^j$;

$\{p_I^j\} = P_I^j \supset P^j$.

Then

1. $(\exists\ p,p' \in SUCC(s))\ (\exists\ k \in \{1,\ldots,l\}: k \neq j): p \in P^j,\ p' \in P^k$
2. $(\exists\ sepa \in SEPA_{j,k}): sepa = (q,s,p')$
3. $POST(s,p) = 1$, $POST(s,p') = 1$ if $\{p'\} \supset P^k$, $POST(s,p') = n \in \mathbb{N}$ if $\{p'\} = P^k$
4. $(\forall ipa \in IPA_j: ipa = (p_I^j,\ldots))\ (\forall\ ipa' \in IPA_k: ipa' = (p',\ldots)): V(ipa) \cap V(ipa') =$

Proof:

1. By def.3 and lemma 3, $(\exists\ p_F^j \in P_F^j \supset P^j)\ (\exists\ sp_F$ simple path$): sp_F = (q,s,\ldots,p_F^j)$.
   By def.4 and def.5, $p' = p_I^k \in P^k$.
2. By def.8, $(q,s,p') \in SEPA_{j,k}$
3. By def.4, $POST(s,p) = 1$, and by def.5, $POST(s,p') = 1$ or $n \in \mathbb{N}$.
4. As by def.5, $P^j \cap P^k = \emptyset$ and $T^j \cap T^k = \emptyset$, by def.7 $V(ipa) \cap V(ipa') = \emptyset$.

■

Now it will be shown that for any $epa \in EPA_{j,k}$ there exists a s-transition where the component j is left. Furthermore, the component k can be entered only via its initial place or a r-transition.

By the statements of the next four lemmata it should become clear how the "interconnections" between different components look like and that it is not possible to represent a 'GOTO' from one task into another one by an Extended Control Structure Net.

As relocating the control flow from one task into another one would generally lead to unpredictable results of a parallel program this is considered to be an error in the mentioned programming languages anyway.

Lemma 5:

Let $gecsn \in GECSN$, $gecsn = (\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i; PRE, POST)$;

$j \neq k \in \{1,\ldots,l\}$; $T^j \neq \emptyset$; $p \in P^j$; $q \in P^k$;

$epa \in EPA_{j,k}$: $epa = (p,\ldots,q)$.

Then

1. $(\exists\ s \in S: s \in T^j)\ (\exists\ p' \in SUCC(s): s \notin P^j)$: $epa = (p,\ldots,s,p',\ldots,q)$

2. $epa = (p,\ldots,s,p',\ldots,p_I^k,\ldots,q)$ or

   $(\exists\ r \in R: r \in T^k)\ (\exists\ q' \in PRED(r): q' \notin P^k)$: $epa = (p,\ldots,s,p',\ldots,q',r,\ldots,q)$

Proof:

1. As $T^j \neq \emptyset$, $\{s\} = U^j$ according to def.5 (iv).

2. By def.5 (iv), $(\forall p'' \in P^k: p'' \neq p_I^k)(\forall t \in \bigcup_{i=1}^{l} T^i): POST(t,p'') = 0$

   or

   by def.5 (iii), $(\exists q' = P^m: m \neq j,k)(\exists t \in T^k: PRE(q',t) = n \in \mathbb{N}): t = r$

Lemma 6:

Let $gecsn \in GECSN$, $gecsn = (\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i; PRE, POST)$;

$j \in \{1,\ldots,l\}$; $T^j \neq \emptyset$; $p \in P^j: |SUCC(p)| > 1$.

Then

$(\forall t \in SUCC(p)): t \in T^j$

Proof:

By contradiction: $(\exists t \in SUCC(p)): t \notin T_j$

Then $(\exists k \neq j \in \{1,\ldots,l\})\ (\exists t \in T^k)\ (\exists sepa \in SEPA_{j,k}): sepa = (p,t,q)$

contradicting lemma 5.

Lemma 7:

Let gecsn $\in$ GECSN, gecsn = $(\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i;$ PRE, POST);

$j \in \{1,\ldots,l\}$; $T^j \neq \emptyset$; $p \in P^j$: PRED(p) > 1 .

Then

$(\forall t \in \text{PRED}(p))$: $t \in T^j$

Proof:

By contradiction: $(\exists\ t \in \text{PRED}(p))$: $t \notin T^j$

Then $(\exists\ k \in \{1,\ldots,l\} : k \neq j)$ $(\exists\ t \in T^k)$ $(\exists\ \text{sepa} \in \text{SEPA}_{k,j})$: sepa = (q,t,p)

contradicting lemma 5.

Lemma 8:

Let gecsn $\in$ GECSN, gecsn = $(\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i;$ PRE, POST);

$j \in \{1,\ldots,l\}$; $\{p\} = P^j$.

Then

1. $(\forall t \in \text{PRED}(p))$ $(\exists\ k \neq j \in \{1,\ldots,l\})$: $t \in T^k$ and $t \in S$
2. $(\forall t \in \text{SUCC}(p))$ $(\exists\ k \neq j \in \{1,\ldots,l\})$ $t \in T^k$ and $t \in R$

Proof:

$\{p\} = P^j$: then by def.3, $T^j = \emptyset$, thus

1. (i) $(\forall t \in \text{PRED}(p))(\exists k \neq j \in \{1,\ldots,l\})$: $t \in T^k$

   (ii) As t T^k, by def.3, $(\exists\ p' \in \text{SUCC}(t))$: $p' \in P^k$. Thus $t \in S$

2. (i) $(\forall$ t SUCC(p))$(\exists k \neq j \in \{1,\ldots,l\})$: $t \in T^k$

   (ii) As $t \in T^k$, by def.3, $(\exists\ p' \in \text{PRED}(t))$: $p' \in P^k$. Thus $t \in R$

The following lemma assures the existence of a non-alternating simple path (definition 9) from the initial place of a graph of an Extended Control Structure Net to an arbitrary place.

Lemma 9:

Let $gecsn \in GECSN$, $gecsn = (\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i; PRE, POST)$;

$j, n-1, n \in \{1,\ldots,l\}$; $q \in P^n$.

Then

$(\exists\, nasp \in NASP^{1}_{p_I^1,q})$: $nasp = (p_I^1,\ldots,q)$

Proof:

By induction:

(i) $q \in P^1$: then $(\exists$ simple path $sp=(p_I^1,\ldots,q))$: $sp \in NASP^{1}_{p_I^1,q}$

(ii) Now assume that (after an eventual renumbering of the components) vertices of the components $1,\ldots,n-1$ are contained in $nasp_{n-1} \in NASP^{1}_{p_I^1,p}$: $nasp_{n-1} = (p_I^1,\ldots,p)$, such that $(\exists\, s \in T^{n-1}: s \in SUCC(p))$: $POST(s,p_I^n) = n \in \mathbb{N}$.

(iii) By def.3, $(\forall q \in P^n)$ $(\exists$ simple path $sp)$: $sp = (p_I^n,\ldots,q)$.

Clearly, $sp \in NASP^{n}_{p_I^n,q}$.

Then compose $nasp_n$ by the concatenation of $nasp_{n-1}$, s, sp.

Thus, $nasp_n = (p_I^1,\ldots,p,s,p_I^n,\ldots,q)$.

Then $(\forall t \in T^j: t \in V(nasp_n))$ $(\forall q' \in PRED(t): q' \in V(nasp_n))$: $q' \in P^j$.

Thus, $nasp_n \in NASP^{1}_{p_I^1,q}$

■

3. EXTENDED CONTROL STRUCTURE NETS

In this section there is finally the concept of the dynamical behaviour of a Petri net introduced (/18/). By adding a "marking" to the definition of graphs of Petri nets it is possible to model dynamical (discrete) concurrent systems such as parallel programs.

The remaining part of this paper is devoted to the question of "liveness" of Extended Control Structure Nets comparable to the "proper termination" of parallel programs. It is shown that the analysis of the structural properties of an Extended Control Structure Net leads to necessary and sufficient conditions for the dynamical liveness property.

This result makes it possible to apply the following procedure to the analysis of a parallel program for proper termination (absence of deadlocks):

- Represent the control structure of a parallel program by a graph of an Extended Control Structure Net,
- apply the algorithms based on the seven liveness conditions.

If this worst-case analysis gives the result that all the conditions are satisfied it is proved at compile-time that the program will always have a proper termination.

3.1. Basic notations and definitions

Definition 10:

Let $gecsn \in GECSN$, $gecsn = (\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i; PRE, POST)$.

1.(i) A marking is a mapping $m: \bigcup_{i=1}^{l} P^i \rightarrow \mathbb{N} \cup \{0\}$

(ii) An initial marking is denoted by m_I

2. $t \in \bigcup_{i=1}^{l} T^i$ is called activated at a marking m iff

$(\forall p \in PRED(t)) : m(p) \geq PRE(p,t)$

3. $t \in \bigcup_{i=1}^{l} T^i$, t activated at m, fires according to the firing rule, generating a new marking m':

(i) $(\forall p \in PRED(t))$: $m'(p) = m(p) - PRE(p,t)$

(ii) $(\forall q \in SUCC(t))$: $m'(q) = m(q) + POST(t,q)$

Notation: $m \,[t\rangle\, m'$

4. A marking m_k $(k > 0)$ is called reachable from a marking m_o iff

$(\exists\, t_1,\ldots,t_k \in \bigcup_{i=1}^{l} T^i)$ $(\exists$ markings $m_1,\ldots,m_{k-1})$: $m_o\,[t_1\rangle\, m_1, m_1\,[t_2\rangle\, m_2, \ldots, m_{k-1}\,[t_k\rangle\, m_k$

Notation: $m_k \in [m_o]$ (m_k belongs to the reachability class of m_o)

Definition 11:

1. ecsn = (gecsn; m_I) is called Extended Control Structure Net iff

(i) $gecsn \in GECSN$: $gecsn = (\bigcup_{i=1}^{l} P^i, \bigcup_{i=1}^{l} T^i; PRE, POST)$

(ii) $m_I(p) = \begin{cases} 1 & \text{if } p = p_I^1 \\ 0 & \text{else} \end{cases}$

2. $ECSN := \{ ecsn = (gecsn; m_I) \mid ecsn \text{ is Extended Control Structure Net} \}$

Now, the liveness of Extended Control Structure Nets is defined and explaine by an example (fig.8).

Definition 12:

Let $ecsn \in ECSN$; $j \in \{1,\ldots,l\}$; $r \in R: r \in T^j$, $q \in PRED(r): q \in P^j$.

ecsn is called live iff

$(\forall m \in [m_I\rangle: m(q)=k \geq 1)(\exists m' \in [m\rangle: m'(q)=k \geq 1)(\forall q' \neq q \in PRED(r)): m'(q') \geq PRE(q',r)$

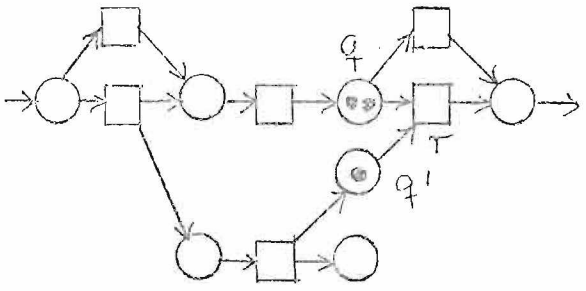


Fig.8: Example

It follows from definition 12, that in a live Extended Control Structure Net at a marking $m(q) \geq 1$, there must exist a marking m' reachable from m such that r as well as t are activated at m'.

There is an obvious interpretation of this definition: as places like q are representing decision statements in programs, each of the outgoing branches may be executed depending of the result of the performed test.

It becomes clear now, too, why the liveness analysis is a worst-case analysis: all the tests are assumed to be independent from each other as the representing places do not contain any interpretation.

By the following lemma, the existence of a marking for each place in an Extended Control structure Net is shown.

Lemma 10:

Let $ecsn \in ECSN$, live;

$j,k \in \{1,\ldots,l\}$; $q \in P^j$, $p \in P^k$;

$nasp \in NASP^1_{p_I,q}$: $nasp = (p_I^1,\ldots,q)$, $t \in \bigcup_{i=1}^{l} T^i$: $t \in V(nasp)$;

$nasp' \in NASP_{p,q}$: $nasp = (p,\ldots,q)$, $t' \in \bigcup_{i=1}^{l} T^i$: $t' \in V(nasp')$.

Then

1. $(\exists\, m \in [m_I])\ (\forall\, q' \in PRED(t))$: $m(q') \geq PRE(q',t)$

2. $(\exists\, m' \in m_I\ )\ (\forall q'' \in PRED(t'))$: $m'(q'') \geq PRE(q'',t')$

Proof:

1. By contradiction:

   $(\exists\, nasp = (p_I^1,\ldots,q))\ (\exists\, t \in V(nasp))\ (\forall\, m \in [m_I])\ (\exists\, q' \in PRED(t))$: $m(q') < PRE(q',t$

   Assume t to be the first transition on nasp having this property.

   Then $p' \in PRED(t)$: $p' \in V(nasp)$ is marked at m and by def.9, belongs to the same component as t.

   Thus, $(\exists\, m \in [m_I]: m(p') \geq 1)\ (\forall\, \bar{m} \in [m])\ (\exists\, q' \in PRED(t))$: $\bar{m}(q') < PRE(q',t)$

   contradicting def.12.

2. $nasp' \in NASP_{p,q}$:

   then $(\exists\, nasp'' \in NASP^1_{p_I,q}$: $nasp'' = (p_I^1,\ldots,p,\ldots,q)$ containing nasp' as subpath.

   Thus, the proof of part 1 can be applied to nasp" .

The following lemma gives a basic property of live Extended Control Strucure Nets concerning the existence of certain non-alternating simple paths in respect to a r-transition.

Lemma 11:

Let ecsn $\in$ ECSN, live;

$j,k,m \in \{1,\ldots,l\} : j \neq k;$

$r \in T^j: r \in R;\ q,q' \in PRED(r): q \in P^j,\ \{q'\} = P^m;$

$s \in T^k: s \in S;\ p' \in PRED(s): p' \in P^k;$

$s' \in S;$

$nasp \in NASP^{1}_{p_I,q}: nasp = (p_i^1,\ldots,q)$.

Then

$$(\exists\, s \in V(nasp))\ (\exists\, nasp' \in NASP_{p',q'}: nasp' = (p',s,\ldots,s',q')): \sum_{s' \in \bigcup_s V(nasp')} POST(s',q') \geq PRE(q',r)$$

Proof:

By contradiction:

$$(\exists\, nasp \in NASP^{1}_{p_I,q})\ (\forall\, s \in V(nasp))\ (\forall\, nasp' \in NASP_{p',q'}): \sum_{s'} POST(s',q') < PRE(q',r)$$

By lemma 11.1, $(\exists\, m \in [m_I]): m(q) \geq 1;$

by lemma 8, $(\forall\, t \in PRED(q')): t \in S;$

by lemma 11.1, each s on nasp has fired when m is reached, thus all the initial places $p_I \in SUCC(s)$ are marked, and by lemma 11.2, each s' can fire.

But as $\sum_{s'} POST(s',q') < PRE(q',r)$ there is a contradiction to def.12.

Corollary 1:

$(\exists\, nasp \in NASP^{1}_{p_I,q}: nasp = (p_I^1,\ldots,q)\ (\forall\, s \in V(nasp)): NASP_{p',q'} = \emptyset$

Then

ecsn is not live

Corollary 2:

$(\exists\, nasp \in NASP^{1}_{p_I,q})\ (\forall\, s \in V(nasp))\ (\forall\, nasp' \in NASP_{p',q'}): \sum_{s'} POST(s',q') \not< PRE(q',r)$

Then

ecsn is not live

The following definition 13 reflects the properties a place has to have if it is to represent a semaphore. Usually, semaphores are used in order to synchronize the sharing of resources which implies that there are not enough resources to satisfy all requests. In fact, if there are sufficient many resources provided for the "users" there doesn't arise any new problems. According to these facts, only these places are called semaphore places, where there cannot exist a marking such that all the r-transitions connected to such a place and modeling the P-operation (/4/) can be enabled at one time. The non-existence of that marking is expressed by statical properties avoiding to have to look for all possible markings reachable from the initial marking in order to be able to the determination of semaphore places. In fig.9, examples for this definition are given.

Definition 13:

Let $ecsn \in ECSN$;

$j,k,m \in \{1,\ldots,l\}$: $j \neq m$;

$\{x\} = P^m$, $R_x := \{ r_1,\ldots,r_{j_x} \in R \mid r \in SUCC(x) \text{ and } j_x \geq 2 \}$

$q_i \in PRED(r_i)$: $r_i \in T^k \rightarrow q_i \in P^k$ $\quad (i = 1,\ldots,j_x)$; $s_i, s' \neq s'' \in S$;

$nasp_i \in NASP_{p_I^1, q_i}$: $nasp_i = (p_I^1,\ldots,p_i',s_i,\ldots,q_i)$,

$nasp_i' \in NASP_{p_i',x}$: $nasp_i' = (p_i',s_i,\ldots,s_i',x)$; $\bar{p} \in V(nasp')$: $|SUCC(\bar{p})| > 1$, $t \neq t' \in SUCC(\bar{p})$

$NASP'_{p_i',x} := NASP_{p_i',x} \setminus \{ \bar{nasp}_i' \in NASP_{p_i',x} : \bar{nasp}_i' = (p_i',s_i,\ldots,\bar{p},t,\ldots,s'',x) \mid$

$(\exists\, nasp_i' = (p_i',s_i,\ldots,\bar{p},t',\ldots,s',x)) \}$

1. x is called semaphore place iff

(i) $(\forall r \neq r' \in R_x)$: $r \in T^j \rightarrow r' \notin T^j$ and $(\exists\, nasp_i \in NASP_{p_I^1, q_i} : i=1,\ldots,j_x)$

$(\exists \bigcup_{i=1}^{j_x} NASP'_{p_i',x})$: $\sum_{s_i' \in V(nasp_i'):\ nasp_i' \in \bigcup_{i=1}^{j_x} NASP'_{p_i',x}} POST(s_i',x) < \sum_{i=1}^{j_x} PRE(q_i,r_i)$

or

(ii) $(\forall r \in R_x)$: $r \in T^j$ and

$(\exists\, sipa \in SIPA_j : sipa = (p_I^j,\ldots,q_1,r_1,\ldots,q_2,r_2,\ldots,q_{j_x}))$

$(\exists\, nasp \in NASP_{p_I^1, q_1})$: $\sum_{s' \in V(nasp'):\ nasp' \in NASP'_{p',x}} POST(s',x) < \sum_{i=1}^{j_x} PRE(q_i,r_i)$

2. $SEMAP := \{ x \in \bigcup_{i=1}^{l} P^i \mid x \text{ satisfies condition (i) or (ii)} \}$

In order to give an easier and more understandable formulation of the liveness conditions, the notation of (s,q')-complete non-alternating simple paths is introduced now. This property turns out to appear in all the seven conditions. Examples are given in fig.10 and fig.11.

Definition 14:

Let ecsn ECSN;

$j,k,m \in \{1,\ldots,l\}: j \neq m;\quad p \in \bigcup_{i=1}^{l} P^i;$

$r \in T^j: r \in R;\ q \in PRED(r): q \in P^j;\ q' \in PRED(r): \{q'\} = P^m;$

$s,s' \neq s'' \in S;$

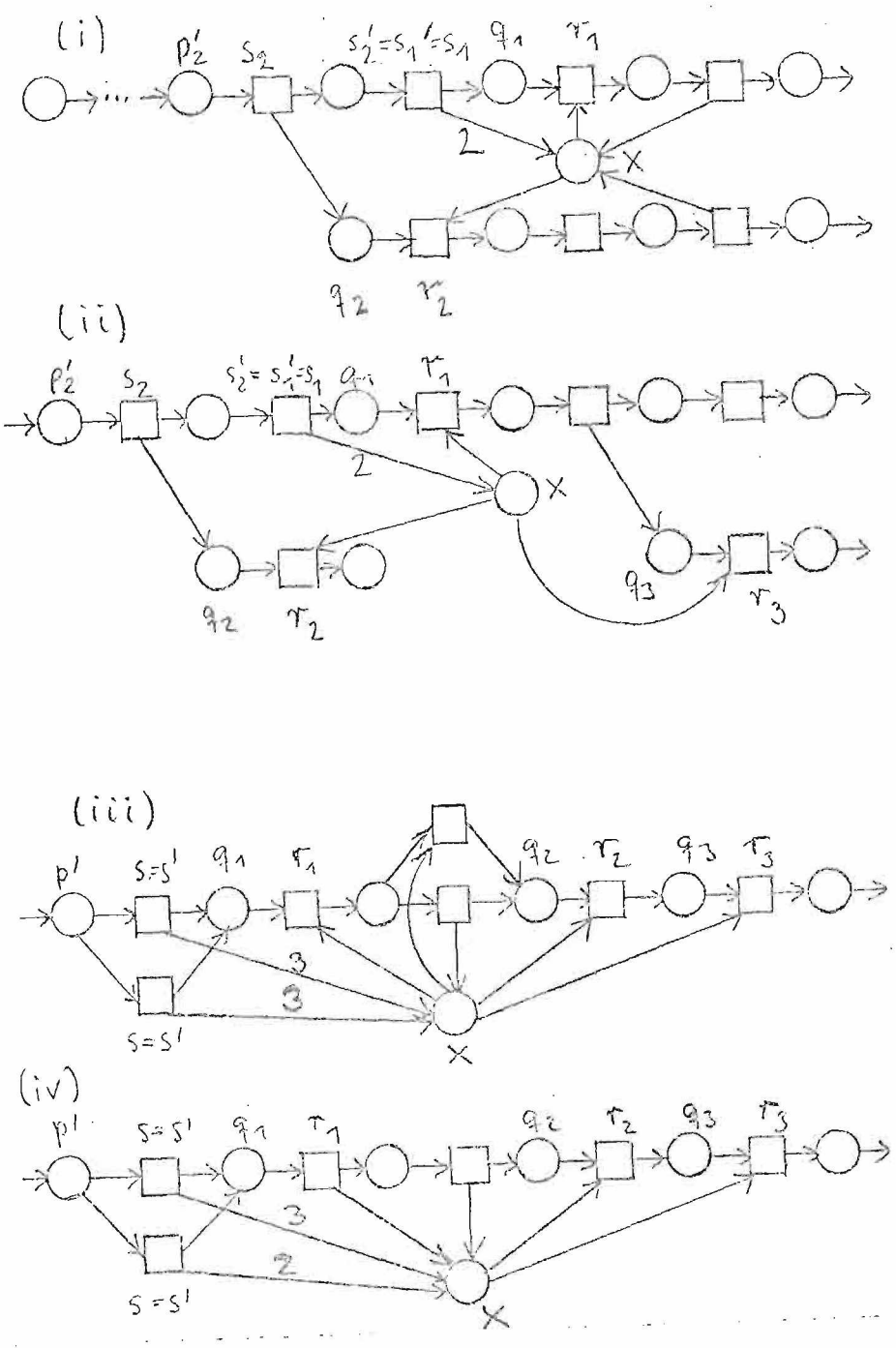


Fig.9: Examples for def.13: x of (i) and (iii) are not semaphore places,
x of (ii) and (iv) are semaphore places

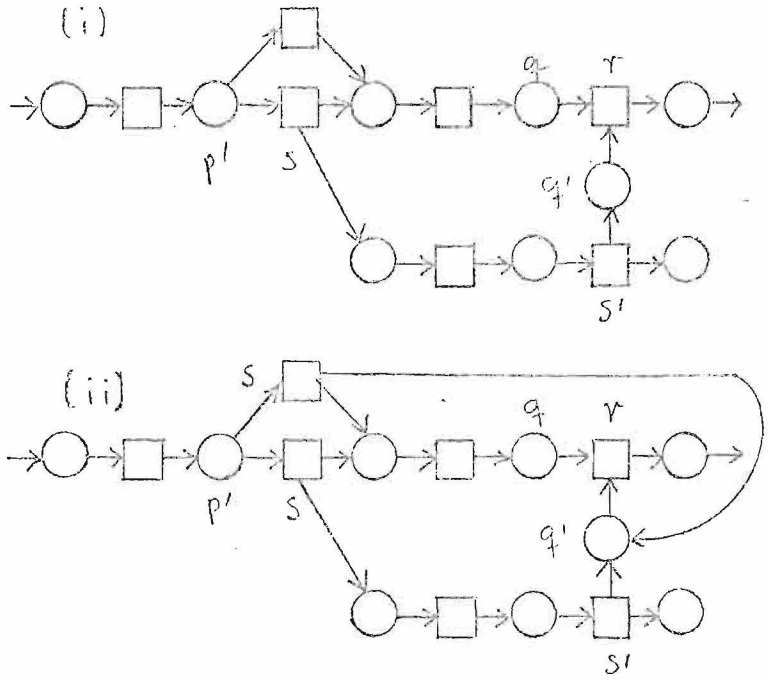


Fig. 10: Illustration of def.14 (i). (i): not (s,q')-complete,(ii):(s,q')-complete

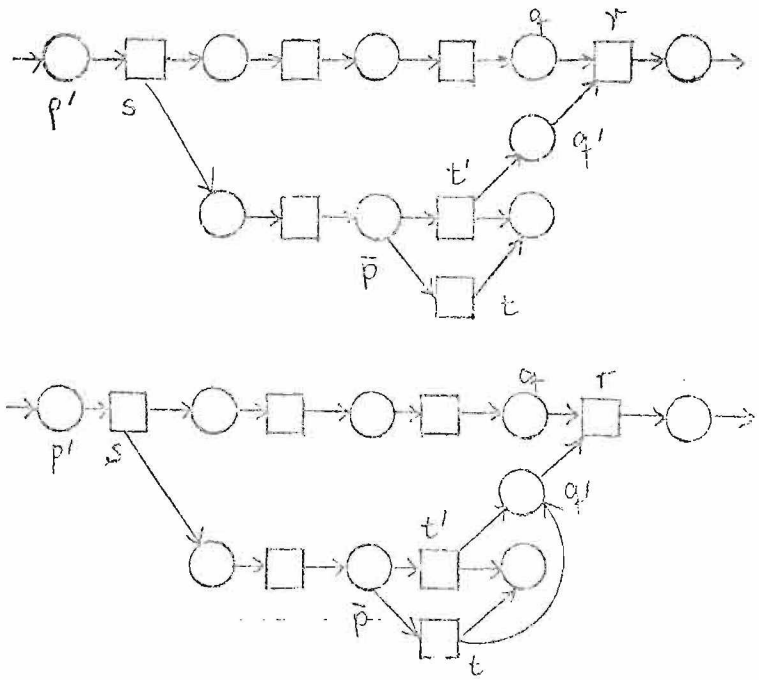


Fig. 11: Illustration of Def.14.(ii). (i):not(s,q')-complete,(ii):(s,q')-complete

3.2. The liveness conditions

In this final section, the seven necessary and sufficient conditions for the liveness of Extended Control Structures Nets are presented in the lemmata 12...18. Examples for each condition are given following the actual lemma.

Lemma 12:

Let $ecsn \in ECSN$, live;

$j,m \in \{1,\ldots,l\} : j \neq m$;

$r \in T^j : r \in R$; $q \in PRED(r): q \in P^j$; $q' \in PRED(r): \{q'\} = P^m$;

$nasp \in NASP_{P_I^1,q}: nasp = (p_I^1,\ldots,q)$.

Then

$$(\forall\, nasp \in NASP_{P_I^1,q}): nasp\ (s,q')\text{-complete} \quad : \quad \sum_{s' \in V_{nasp'}} POST(s',q') \geq PRE(q',r)$$

Proof:

By contradiction:

$$ecsn\ live, (nasp\ not\ (s,q')\text{-complete}) : \sum_{s' \in V_{nasp'}} POST(s',q') < PRE(q',r)$$

Case 1.

By def.14 (i),

$$((\exists nasp)\ (\forall\, s \in V(nasp))\ (\forall\, nasp'\ NASP_{p',q'}): nasp' \neq (p',s,\ldots,q')): \sum_{s' \in V_{nasp'}} POST(s',q') < PRE(q',r)$$

By corollary 2, this contradicts def. 12 (see fig.12).

Case 2:

By def.14 (ii),

$$((\exists\, \bar{p} \in V(nasp'))\ (\exists\, \bar{t} \in SUCC(\bar{t}))\ (\forall\, \bar{n}asp' \in NASP_{p',q'}):\ \bar{n}asp' \neq (p', s, \ldots, \bar{p}, \bar{t}, \ldots, q')):$$

$$\sum_{s' \in V_{nasp'}} POST(s', q') < PRE(q', r)$$

contradicting def.12 by corollary 2 (see fig.13).

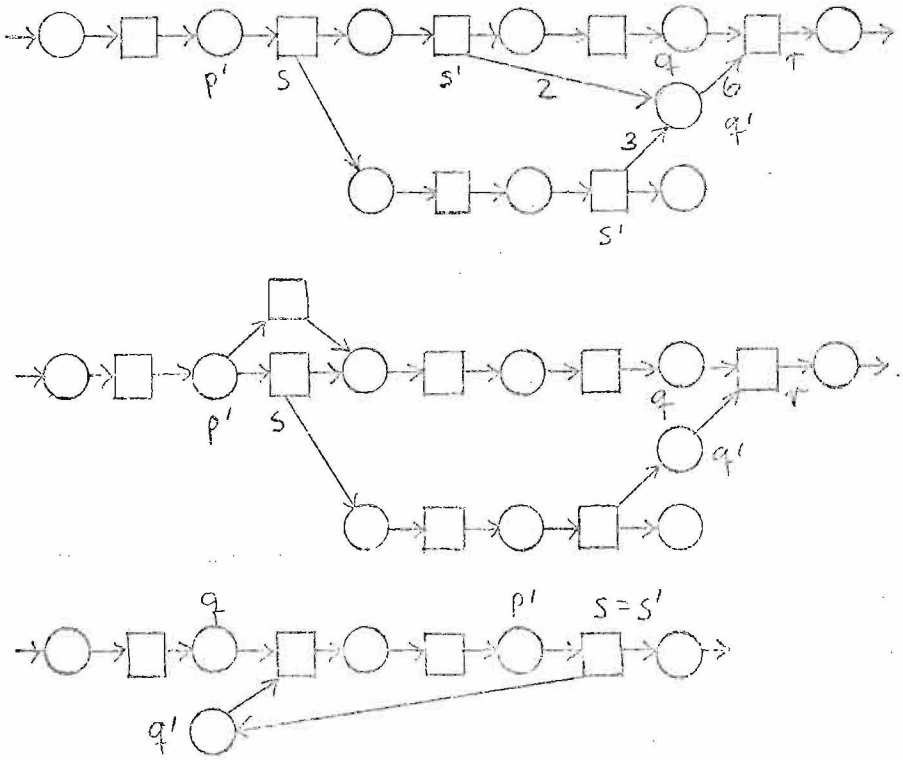


Fig.12: Configurations of case 1

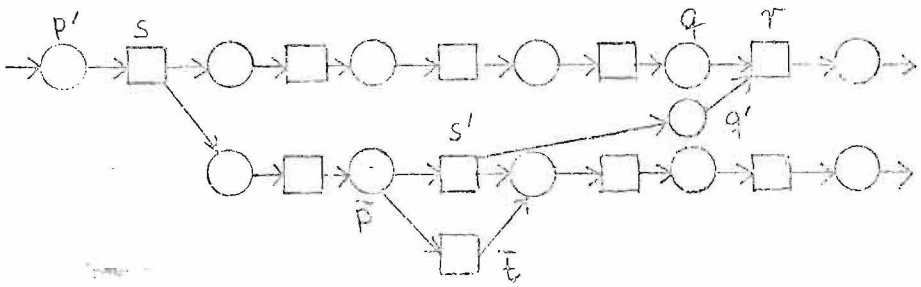


Fig.13: Configuration of case 2

Lemma 13:

Let ecsn $\in$ ECSN, live;

$j,m \in \{1,\ldots,l\}$: $j \neq m$;

$r \in T^j$: $r \in R$; $q \in PRED(r)$: $q \in P^j$; $q' \in PRED(r)$: $\{q'\} = P^m$;

$nasp \in NASP_{p_I^1,q}$: $nasp = (p_I^1,\ldots,q)$;

$sct \in SCT_m$: $sct = (q',\ldots,q')$.

Then

$(\exists\, r \in V(sct))\ (\forall\, NASP')\ (\forall\, nasp' \in NASP'$: $nasp'=(p',s,\ldots,s',q'))$: $V(nasp') \cap V(sct) = \{q'\}$

Proof:

By contradiction:

$(\forall\, r \in V(sct))\ (\exists\, NASP')\ (\exists\, nasp' \in NASP')$: $V(nasp') \cap V(sct) \neq \{q'\}$

As $q' \in V(sct)$ and $q' \in V(nasp')$, $V(nasp') \cap V(sct) \supset \{q'\}$.

By def.8, at least $\{s',q'\} \supset V(nasp') \cap V(sct)$.

Thus, s' cannot fire as no transition of sct can fire and because the condition of lemma 12 holds, too, there is no transition s' which still could be activated, contradicting def.12 (see fig.14).

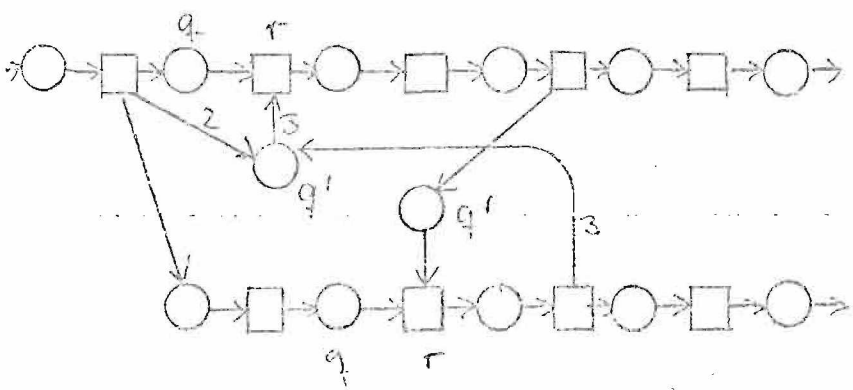


Fig.14: Configuration of the negation of lemma 13

Lemma 14:

Let ecsn $\in$ ECSN, live;

$j,m \in \{1,\ldots,l\} : j \neq m;$

$r \in T^j : r \in R;\quad q \in PRED(r) : q \in P^j;\quad q' \in PRED(r) : \{q'\} = P^m;$

$lp \in LP_j : lp = (p,\ldots,q,r,\ldots,p)$.

Then

$$(\forall lp \in LP_j : lp = (p,\ldots,q,r,\ldots,p)) : lp\ (s,q')\text{-complete} : \sum_{s' \in V_{nasp'}} POST(s',q') \geq PRE(q',r)$$

Proof:

By contradiction (cf. proof of lemma 12):

$$(\exists\ lp \in LP_j) : lp \text{ not } (s,q')\text{-complete} : \sum_{s'} POST(s',q') < PRE(q',r)$$

By lemma 10, $(\exists\ m_1 \in [m_I] : m_1(q) \geq 1$, and by lemma 12, $m_1(q') = k_o \geq PRE(q',r)$.

Thus, r can fire and by lemma 10, $(\exists\ m_2 \in [m_1]) : m_2(p) \geq 1$ and $(\exists\ m_3 \in [m_2]) : m_3(q) \geq 1$.

At each traversal of lp, the number of tokens k_o on q' will be decreased by

$PRE(q',r) + POST(s',q') > 0$.

Thus, after a finite number of traversals of lp, $k_o < PRE(q',r)$ contradicting def.12 (see fig.15) .

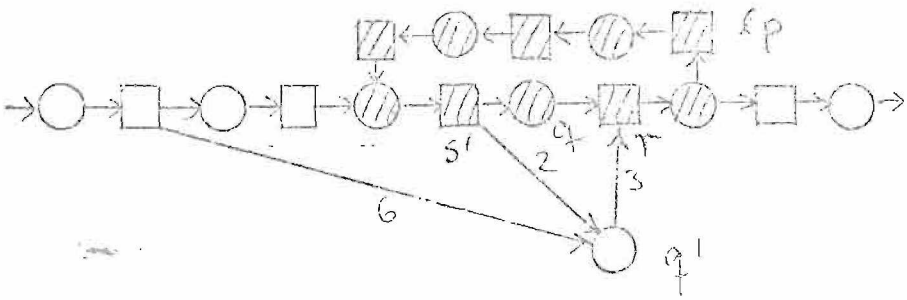
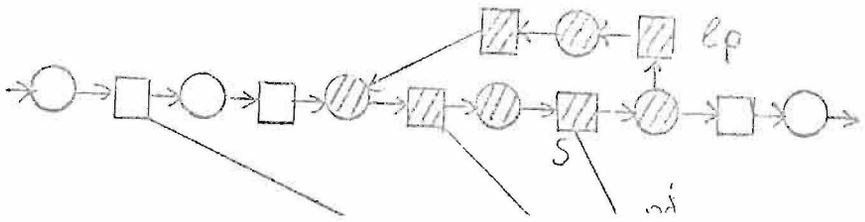


Fig.15: The condition of lemma 14 is not satisfied



Lemma 15:

Let ecsn $\in$ ECSN, live;

$j,k,m \in \{1,\ldots,l\}$: $j \neq m$;

$r \in T^j$: $r \in R$; $q \in PRED(r)$: $q \in P^j$; $q' \in PRED(r)$: $\{q'\} = P^m$; $q'' \in SUCC(r)$: $q'' \in P^j$;

$lp \in LP_k$; $s \in S$: $s \in V(lp)$; $p_1^j \in SUCC(s)$;

$s_1, s_1', s_2, s_2' \in S$.

Then

$(\forall lp: s \in V(lp))\ (\forall sipa \in SIPA_j : sipa = (q'',\ldots,p_F^j))$:

lp (s_1,q')-complete or sipa (s_2,q')-complete : $\sum_{s_1' \in V_{nasp_1'}} POST(s_1',q') + \sum_{s_2' \in V_{nasp_2'}} POST(s_2',q') \geq$

$\geq PRE(q',r)$

Proof:

By contradiction (cf. proof of lemma 12):

$(\exists lp: s \in V(lp))(\exists sipa \in SIPA_j)$: lp not (s_1,q')-complete and sipa not (s_2,q')-complete:

$\sum_{s_1'} POST(s_1',q') + \sum_{s_2'} POST(s_2',q') < PRE(q',r)$

First assume without loss of generality, that lp is k_q times traversed: then clearly,

$(\exists m \in [m_F]: m(q) = k_q$.

According to def.12, r must be able to fire k_q times, i. e. $k_q \cdot PRE(q',r)$ tokens are required to make that possible.

Available tokens are:- a constant number of tokens by lemma 12 which need not to be considered by choosing k_q correspondentily and by the same arguing as in the proof of lemma 14.

- $k_q \cdot (\sum_{s_1'} POST(s_1',q') + \sum_{s_2'} POST(s_2',q')) < k_q \cdot PRE(q',r)$

Thus, r cannot fire k_q times contradicting def.12 (see fig.16) .

Proof:

By contradiction (cf. proof of lemma 12) :

$(\exists\ nasp_{i+1} \in NASP^{1}_{p_I, q_{i+1}})$: nasp not (s,x)-complete: $\sum_{s'} POST(s',x) < \sum_{k} PRE(x,r_k)$

By lemma 10, r_i can fire and $(\exists\ m \in [m_I] : m(q_{i+1}) \geq 1$.

Up to now, $\sum_{k=1}^{i} PRE(x,r_k)$ tokens are consumed by the firing of $r_1,\ldots,r_i$ and for the firing of r_{i+1}, additional $PRE(x,r_{i+1})$ tokens are requested being not available by the assumption and def.13 (ii) .

This contradicts def.12 (see fig.17) .

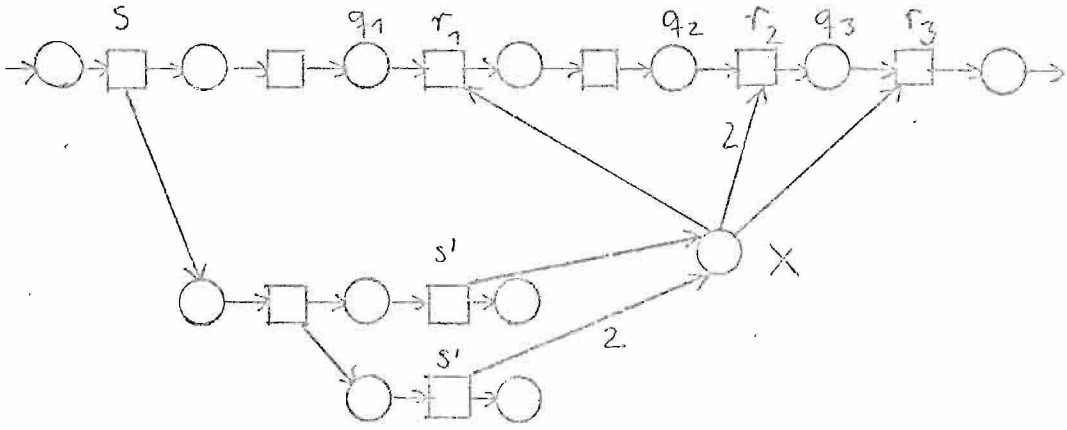


Fig.17: The condition of lemma 16 is not satisfied for i+1 = 3

Lemma 17:

Let $ecsn \in ECSN$, live;

$j,m \in \{1,\ldots,l\} : j \neq m$;

$\{x\} = P^{in}: x \in SEMAP$; R_x is defined as in def.13: $R_x = \{r_1,\ldots,r_{j_x}\}$;

$q_i \in PRED(r_i): r_i \in T^j \rightarrow q_i \in P^j$;

$nasp_i \in NASP^1_{p_I,q_i} : nasp_i = (p^1_I,\ldots,q_i)$.

Then

$$(\forall nasp_i \in NASP^1_{p_I,q_i} : i=1,\ldots,j_x): nasp_i \ (s,x)\text{-complete} : \sum_{s' \in V_{nasp_i}} POST(s',x) \geq \sum_{k=1}^{j_x} PRE(x,r_k)$$

Proof:

By contradiction (cf. lemma 12):

$$(\exists nasp_i \in NASP^1_{p_I,q_i}): nasp_i \text{ not } (s,x)\text{-complete}: \sum_{s'} POST(s',x) < \sum_{k} PRE(x,r_k)$$

All $r_1,\ldots,r_{j_x}$ have to be able to fire concurrently , thus $\sum_{k=1}^{j_x} PRE(x,r_k)$ tokens are required in total.

This is not satisfied by the assumption contradicting def.12 (see fig.18).

□

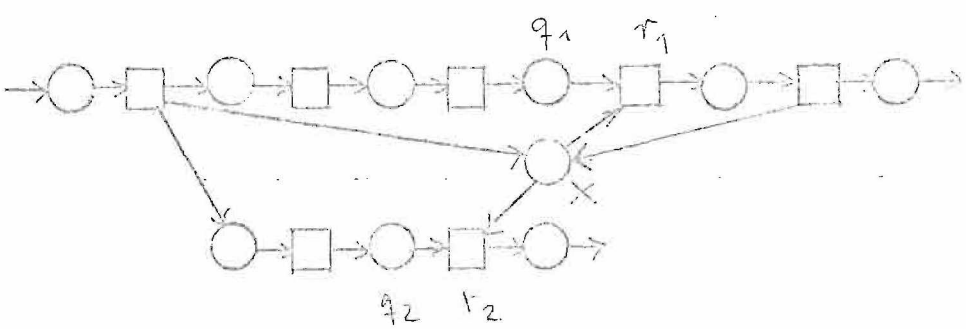


Fig.18: The condition of lemma 17 is not satisfied

Lemma 18:

Let ecsn $\in$ ECSN, live;

$j_1,j_2,j_3,j_4,m_1,m_2 \in \{1,\ldots,l\}$: $j_1,j_3 \neq j_2,j_4$, $j_1,j_2,j_3,j_4 \neq m_1,m_2$;

$r_i \in T^{j_i}$: $r_i \in R$; $q_i \in PRED(r_i)$: $q_i \in P^{j_i}$ $(i=1,\ldots,4)$;

$\{x_1\} = P^{m_1}$, $\{x_2\} = P^{m_2}$: $\{x_1,x_2\} \subseteq SFMAT$;

$r_1 \neq r_4 \in SUCC(x_1)$; $r_2 \neq r_3 \in SUCC(x_2)$;

$nasp_1 \in NASP^1_{p_I^1,q_3}$: $nasp_1 = (p_I^1,\ldots,q_1,r_1,\ldots,q_3)$;

$nasp_2 \in NASP^1_{p_I^1,q_4}$: $nasp_2 = (p_I^1,\ldots,q_2,r_2,\ldots,q_4)$.

Then

$$(\forall\, nasp_1 \in NASP^1_{p_I^1,q_3}): nasp_1(s,x_1)\text{-complete} : \sum_{s' \in V_{nasp_1'}} POST(s',x_1) \geq PRE(x_1,r_1)+PRE(x_1,r_4)$$

or

$$(\forall\, nasp_2 \in NASP^1_{p_I^1,q_4}): nasp_2(s,x_2)\text{-complete}: \sum_{s' \in V_{nasp_2'}} POST(s',x_2) \geq PRE(x_2,r_2)+PRE(x_2,r_3)$$

Proof:

By contradiction (cf. lemma 12):

$$(\exists\, nasp_1 \in NASP^1_{p_I^1,q_3}): nasp_1 \text{ not } (s,x_1)\text{-complete}: \sum_{s'} POST(s',x_1) < PRE(x_1,r_1)+PRE(x_1,r_4)$$

and

$$(\exists\, nasp_2 \in NASP^1_{p_I^1,q_4}): nasp_2 \text{ not } (s,x_1)\text{-complete}: \sum_{s'} POST(s',x_2) < PRE(x_2,r_2)+PRE(x_2,r_3)$$

As it is shown in the proof of lemma 17, neither r_3 nor r_4 can fire contradicting def.12 (see fig.19) .

■

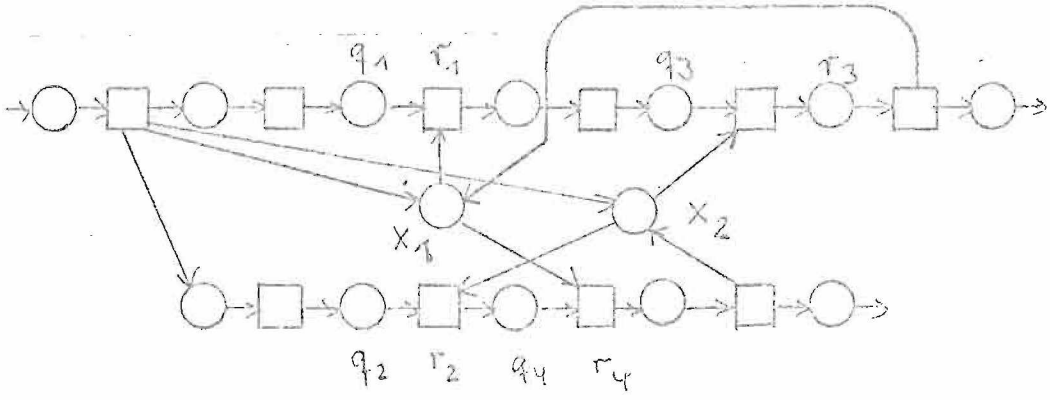


Fig.19: The condition of lemma 18 is not satified

Theorem:

Let ecsn $\in$ ECSN .

ecsn is live iff

the conditions of lemma 12 up to lemma 18 are satisfied.

Proof:

"$\rightarrow$" : see lemma 12...18 .

"$\leftarrow$" :

By contradiction:

The conditions of lemma 12...18 are satisfied and ecsn not live:

$(\exists\, r \in R)\ (\exists\, m \in [m_I\rangle : m(q)=k \geq 1)\ (\forall\, m' \in [m\rangle)\ (\exists\, q' \in PRED(r)) : m'(q') < PRED(q',r)$

Furthermore, let r be the first transition on $nasp \in NASP^1_{p_I,q}$ being not activatable.

CASE A: k = 1:

Case 1: $(\forall lp \in \bigcup_{i=1}^{l} LP_i)$: $r \notin V(lp)$

Case 1.1: $(\forall sct \in \bigcup_{i=1}^{l} SCT_i)$: $r \notin V(sct)$

Case 1.1.1: $(\exists nasp \in NASP^{1}_{p_I,q})$ $(\forall s \in V(nasp))$: $NASP^{}_{p',q'} = \emptyset$

Then nasp is not (s,q')-complete contradicting lemma 12 .

Case 1.1.2: $(\forall nasp \in NASP^{1}_{p_I,q})$ $(\exists s \in V(nasp))$: $nasp' = (p',\ldots,s',q')$

Case 1.1.2.1: $(\exists nasp' \in NASP^{}_{p',q'})$ $(\exists \bar{p} \in V(nasp'))(\exists \bar{t} \in SUCC(\bar{p}))$ $(\forall \bar{n}asp)$: $q' \notin V(\bar{n}asp)$

Then nasp is not (s,q')-complete contradicting lemma 12 .

Case 1.1.2.2: $(\forall nasp' \in NASP^{}_{p',q'})$ $(\forall \bar{p} \in V(nasp'))(\forall \bar{t} \in SUCC(\bar{p}))$ $(\exists \bar{n}asp)$: $q' \in V(\bar{n}asp)$

Case 1.1.2.2.1: $q' \in SEMAP$

Case 1.1.2.2.1.1: Def.13(i) is satisfied for q'

Case 1.1.2.2.1.1.1: The assumptions of lemma 18 are not satisfied for r

Then by lemma 12, at least one $r_i \in R_{q'}$ has fired and a marking m' is reached from m_j such that $(\forall m'' \in [m'])$: $m''(q') < PRE(q',r)$.

It follows that $(\exists nasp_i \in NASP^{1}_{p_I,q_i})$: $\sum_{s' \in V_{nasp_i}} POST(s',q') < \sum_{k=1}^{j_x} PRE(q',r_k)$ contradicting lemma 17 .

Case 1.1.2.2.1.1.2: The assumptions of lemma 18 are satisfied for $r := r_3$

Then $(\exists nasp_1 \in NASP^{1}_{p_I,q_3})$: $nasp_1 = (p^1_I,\ldots,q_1,r_1,\ldots,q_3)$ and

$(\exists nasp_2 \in NASP^{1}_{p_I,q_4})$: $nasp_2 = (p^1_I,\ldots,q_2,r_2,\ldots,q_4)$.

As $x_1,q' \in SEMAP$, let $\bar{n}asp_1 \in NASP^{1}_{p_I,q_1}$, $\bar{n}asp_2 \in NASP^{1}_{p_I,q_2}$ and by def.13 (i),

$$\sum_{s'_i \in V(nasp'):nasp'_i \in \bar{N}ASP'_{p'_1,x_1} \cup NASP'_{p'_2,x_1}} POST(s'_i,x_1) = k_1 < PRE(x_1,r_1) + PRE(x_1,r_4)$$

$$\sum_{s'_i \in V(nasp'):nasp'_i \in \bar{N}ASP'_{p'_2,q'} \cup NASP'_{p'_1,q'}} POST(s'_i,q') = k_2 < PRE(q',r_2) + PRE(q',r_3) .$$

By lemma 17,

$$\sum_{s' \in V_{nasp_1'}} POST(s',q') \geq PRE(q',r_2) + PRE(q',r_3) \quad \text{and}$$

$$\sum_{s' \in V_{nasp_2'}} POST(s',x_1) \geq PRE(x_1,r_1) + PRE(x_1,r_4) \ .$$

Thus, $(\forall\, nasp_3 = (q_1,\ldots,q_3,r_3,\ldots)):nasp_3\ (s_3,x_1)$-complete :

$$\sum_{s_3' \in V_{nasp_3'}} POST(s_3',x_1) + k_1 \geq PRE(x_1,r_1) + PRE(x_1,r_4) \qquad \text{and}$$

$(\forall\, nasp_4 = (q_2,\ldots,q_4,r_4,\ldots)):nasp_4\ (s_4,q')$-complete :

$$\sum_{s_4' \in V_{nasp_4'}} POST(s_4',q') + k_2 \geq PRE(q',r_2) + PRE(q',r_3) \ .$$

Now assume, that there exists s_3 on $nasp_3$ in between q_1 and q_3: then, r_1 and r_2 and, by the firing of s_3, also r_4 and finally r_3=r would fire contradicting this assumption.

The same applies to a s_4 on $nasp_4$ in between q_2 and q_4 .

Thus, neither $nasp_1 = (p_1^1,\ldots,q_1,r_1,\ldots,q_3)$ is (s,x_1)-complete :

$$\sum_{s' \in V_{nasp_1'}} POST(s',x_1) \geq PRE(x_1,r_1) + PRE(x_1,r_4)$$

nor $nasp_2 = (p_1^1,\ldots,q_2,r_2,\ldots,q_4)$ is (s,q')-complete:

$$\sum_{s' \in V_{nasp_2'}} POST(s',q') \geq PRE(q',r_2) + PRE(q',r_3)$$

contradicting lemma 18 .

Case 1.1.2.2.1.2: Def.13 (ii) is satisfied for q'

Then $(\exists\, nasp_{i+1} \in NASP^{1}_{p_I,q_{i+1}})$: $nasp_{i+1} = (p_I^1,\ldots,q_1,r_1,\ldots,q_i,r_i,\ldots,q_{i+1})$

and by the general assumption in the beginning of this proof,

$r_1,\ldots,r_i$ have already fired such that

$\sum_{s' \in V_{nasp_{i+1}}} POST(s',q') < \sum_{k=1}^{i} PRE(q',r_k)$ contradicting lemma 16 .

Case 1.1.2.2.2: $q' \notin SEMAP$

Then $\sum_{s' \in V_{nasp'}} POST(s',q') < PRE(q',r)$ contradicting lemma 12 .

Case 1.2: $(\exists\, sct \in \bigcup_{i=1}^{l} SCT_i): r \in V(sct)$

Then $(\forall\, r \in V(sct))\ (\exists\, nasp \in NASP^{1}_{p_I,q})\ (\exists\, nasp' \in NASP_{p',q'})\ (\forall\, s'' \in V(nasp'): s'' \ V(sct))$

$\sum_{s' \in V_{nasp'} \setminus \{s''\}} POST(s',q') < PRE(q',r)$

contradicting lemma 13.

Case 2: $(\exists\, lp \in \bigcup_{i=1}^{l} LP_i)$: $r \in V(lp)$

Let $lp = (\hat{p},\ldots,q,r,\ldots,\hat{p})$; $nasp \in NASP^{1}_{p_I,q}$: $nasp = (p_I^1,\ldots,q)$.

By lemma 12, $(\exists\, s \in V(nasp))\ (\exists\, nasp' \in NASP_{p',q'})$: $nasp' = (p',s,\ldots,s',q')$

If $s \in V(p_I^1,\ldots,\hat{p})$ the result will be a contradiction to lemma 14.

CASE B: $k > 1$

By def.5, $(\forall\, p_I^i)$: $POST(s',p_I^j) \leq 1$.

Thus the only way of getting multiple tokens into a component is the following:

$(\exists\, lp \in LP_k$: $lp = (\hat{p},\ldots,\hat{p}))$ $(\exists\, \hat{s} \in V(lp) : \hat{s} \subset S)$: $p_I^j \in SUCC(\hat{s})$ and

$(\forall\, r' \in V(lp))$ $(\forall\, q'' \in PRED(r'))$: $NASP_{q,q''} = \emptyset$

Case 1: $(\forall\, sct \in \bigcup_{i=1}^{l} SCT_i)$: $r \notin V(sct)$

Let $lp = (\hat{p},\ldots,\hat{s},\ldots,\hat{p})$; $nasp \in NASP_{p_I^1,q}$: $nasp = (p_I^1,\ldots,\hat{p},\ldots,\hat{s},p_I^j,\ldots,q)$.

By lemma 12, $(\exists\, nasp' \in NASP_{p',q'})$: $\sum_{s' \in V_{nasp'}} POST(s',q') \geq PRE(q',r)$

Case 1.1: $s \in V(p_I^1,\ldots,\hat{p})$ or $s \in V(p_I^j,\ldots,q)$

This contradicts lemma 15.

Case 1.2: $s \in V(\hat{p},\ldots,\hat{s},p_I^j$

Then $\sum_{s' \in V_{nasp'}} POST(s',q') < PRE(q',r)$ contradicting lemma 15 .

Case 2: $(\exists\, sct \in \bigcup_{i=1}^{l} SCT_i)$: $r \in V(sct)$

Then $(\exists\, sipa \in SIPA_j$: $sipa = (q,r,\ldots,p_F^j))$ $(\exists\, s \in V(sipa))$ $(\exists\, nasp' \in NASP_{p',q'})$:

$nasp' = (p',s,\ldots,s',q')$

Case 2.1: sipa is not (s,q')-complete

This contradicts lemma 15 .

Case 2.2: sipa is (s,q')-complete

Then $\sum_{s' \in V_{nasp'}} POST(s',q') < PRE(q',r)$ contradicting lemma 15 .

ACKNOWLEDGEMENTS

The main part of this work was done while the author was as a research associate with the Computer Science Department of the University of Utah at Salt Lake City, Utah, supported by the Deutsche Forschungsgemeinschaft Bonn through grant Me/939/2.

I am indebted to Suhas Patil for the invitation to this Department and for many helpful discussions. The assistance of other staff members, especially of Z. Kohavi and R. M. Keller is thankfully acknowledged, too.

REFERENCES

/1/ P. BRINCH HANSEN: Concurrent Pascal - A Programming Language for Operating System design.-
California Institute of Technology, Information Science, Technical Report No. 10 (April 1974)

/2/ BURROUGHS Corporation: BURROUGHS B 6700/B 7700 ALGOL Language Reference Manual.-
5000649 (1978)

/3/ V. G. CERF: Multiprocessors, Semaphores and a Graph Model of Computation.-
University of California, Computer Science Department, UCLA-10P14-110 (Apr. 1972)

/4/ E. W. DIJKSTRA: Co-operating sequential Processes.-
In: F. GENUYS (Ed.): Programming Languages.- London (1968), p.43-112

/5/ H. J. GENRICH, K. LAUTENBACH: Synchronisationsgraphen.-
Acta Informatica 2 (1973), p. 143-161

/6/ M. H. T. HACK: Extended State Machine allocatable Nets (ESMA) - an Extension of Free Choice Petri Nets Results.-
M.I.T., Project MAC, Computations structures Group Memo 78-1 (1974)

/7/ O. HERZOG, M. YOELI: Control Nets for Asynchronous Systems, Part 1.-
Technion-Israel Institute of Technology, Computer Science Department, TR-74 (May 1976)

/8/ O. HERZOG: Zur Analyse der Kontrollstruktur paralleler Programme mit Hilfe von Petri-Netzen.-
Universität Dortmund, Abteilung Informatik, Bericht Nr. 24/76 (1976)

/9/ O. HERZOG: Automatic Deadlock Analysis of parallel Programs.-
In: E. MORLET, D. RIBBENS (Eds.): International Computing Symposium 1977.-
Amsterdam (1977), p. 209-216

/10/ A. HOLT, F. COMMONER: Events and Conditions.-
Applied Data Research Inc., New York (1970)

/11/ IBM Corporation: OS PL/I Checkout and Optimizing Compilers: Language Reference Manual.-
GC33-0009-3 (1974)

/12/ R. M. KELLER: Generalized Petri Nets as Models for System Verification.-
Princeton University, Department of Electrical Engineering, Technical Report No. 200 (Dec. 1975)

/13/ R. M. KELLER: Formal Verification of parallel Programs.-
Comm. ACM 19,7 (1976), p. 371-394

/14/ H. C. LAUER: Correctness in Operating Systems.-
AD 753122 (1972)

/15/ K. LAUTENBACH: Use of Petri Nets for proving Correctness of concurrent Process Systems.-
Proceedings of the IFIP Congress 74.- Amsterdam (1974), p. 187-191

/16/ R. E. MILLER: Some Relationships between various Models of Parallelism and Synchronization.-
IBM Thomas J. Watson Research Center, RC 5074 (Oct. 1974)

/17/ S. S. PATIL: Limitations and Capabilities of Dijkstra's Semaphore Promitives for Coordination among Processes.-
M.I.T., Project MAC, Computation Structures Group Memo 57 (Feb. 1971)

/18/ C. A. PETRI: Concepts of Net Theory.-
In: Mathematical Foundations of Computer Science, Proceedings of Symposium and Summer School, High Tatras, Sept. 3-8, 1973.-
Mathematical Insitute of the Slovak Academy of Sciences, Computing Research Centre United Nations D.P. Bratislava (1973), p. 137-146

/19/ H. A. SCHMID: An Approach to the Communication and Synchronization of Processes.
In: Proceedings of the International Computing Symposium , Davos 1973.-
Amsterdam (1973), S. 165-171

/20/ M. YOELI: Petri Nets and Asynchronous Control Networks.-
University of Waterloo, Department of Applied Analysis and Computer Science, Research Report CS-73-07 (1973)