Petri Nets 2000

21st International Conference on
Application and Theory of Petri Nets

Aarhus, Denmark, June 26-30, 2000

Introductory Tutorial
Petri Nets

Organised by

Gianfranco Balbo
Jörg Desel
Kurt Jensen
Wolfgang Reisig
Grzegorz Rozenberg
Manuel Silva

June 2000

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF AARHUS
Ny Munkegade, Bldg. 540
DK-8000 Aarhus C, Denmark

Table of Contents

An Informal Introduction
to
Petri Nets

Running Example:
A Producer/Consumer System

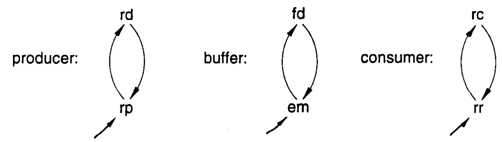
W. Reisig
Humboldt University of Berlin

A Programming Notation

P1: **do** forever
if buffer = empty **then**
buffer := filled
end

P2: **do** forever
if buffer = filled **then**
buffer := empty
end.

A State Based Representation



An Action Based Representation

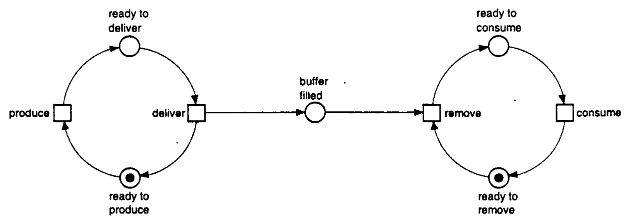
producer = p.d.producer

consumer = r.c.consumer

buffer = $\overline{\mathrm{d}}.\overline{\mathrm{r}}$.buffer

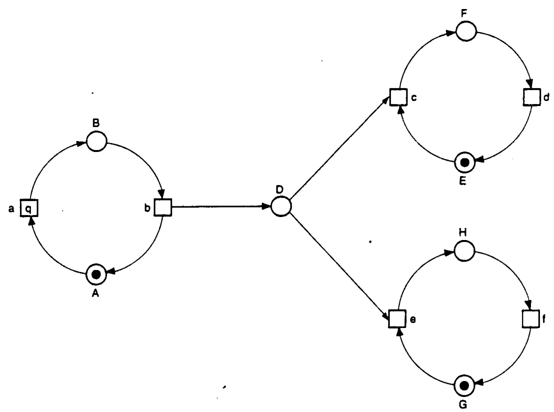
producer || buffer || consumer

Neither State Based Nor Action Based But Well Ballanced



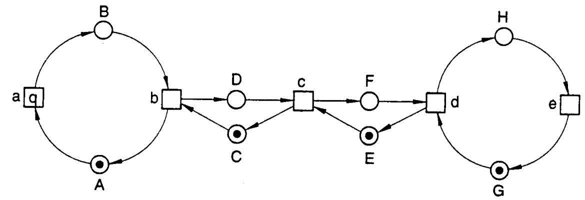
4

Two Consumers



5

Two Buffer Cells



A : ready to produce
B : ready to deliver
C : first buffer cell empty
D : first buffer cell filled

a : produce
b : deliver

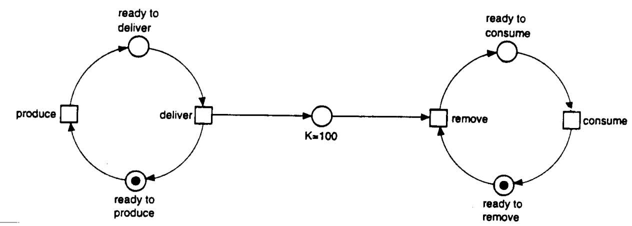
E : second buffer cell empty
F : second buffer cell filled
G : ready to remove
H : ready to consume

d : remove
e : consume

so far: elementary
net
systems

6

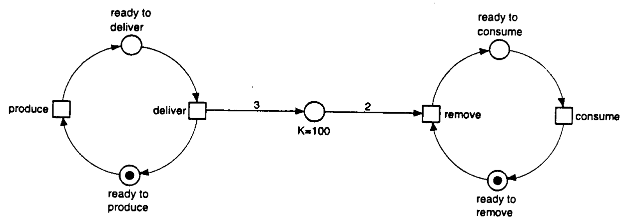
What About 100 Buffer Cells?



... a place/transition net

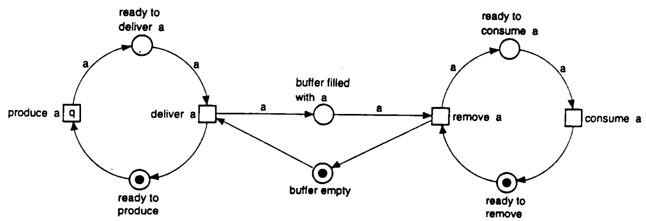
7

Arc Weights of Place/Transition Nets



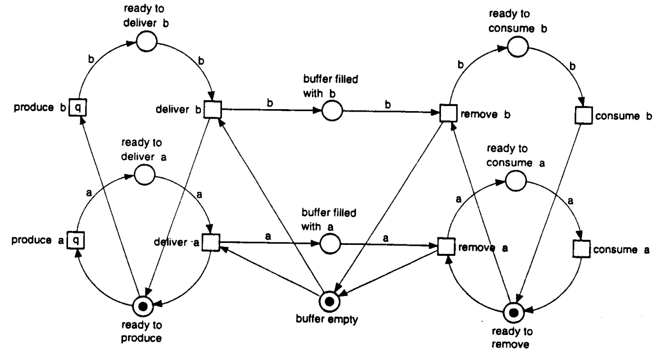
so far: place/transition nets

Producing and Consuming Objects of Sort a

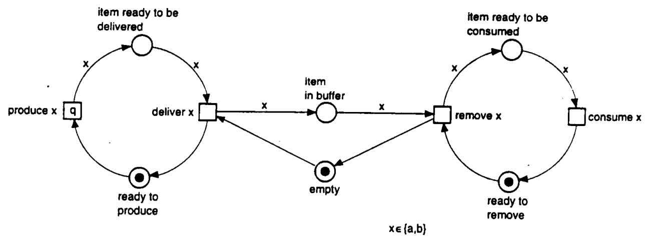


. . . a high-level net

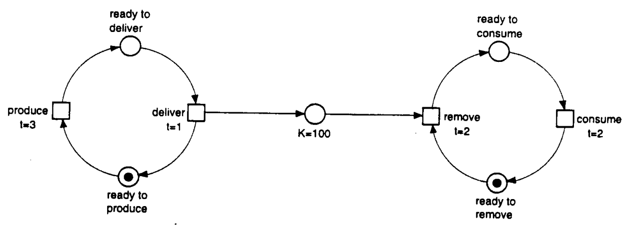
Producing and Consuming Objects of Sort a or b



Producing and Consuming Any Kind of Items



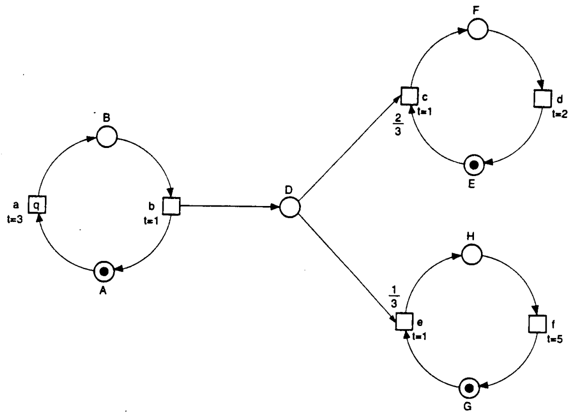
How Long Does It Take to Produce One Item?



... a timed Petri net

12

Stochastic Conflict Resolution



... a stochastic Petri net

13

hence the structure of the tutorial:

G. Rozenberg:		
	Elementary Net Systems	I today II tomorrow
J. Desel / M. Silva:		
	Place / Transition Nets	I today II tomorrow
K. Jensen:		
	High Level Nets	I today II tomorrow
S. Donatelli:		
	Timed and Stochastic Nets	I tomorrow II tomorrow

14

ELEMENTARY NET SYSTEMS

G. Rozenberg
Leiden University, The Netherlands
&
Univ. of Colorado at Boulder, USA

The area of Petri Nets was initiated by C.A. Petri in early 60's.
The chief attraction of this area is the way in which the basic aspects of distributed systems are identified both conceptually and mathematically.

In our lecture we will illustrate this point using the most fundamental class of Petri nets called elementary net systems.

The guiding principles of net theory in formulating the basic notions of states and changes-of-states (called transitions) are:

1) States and transitions are two intertwined but distinct notions that deserve an even-handed treatment.

2) Both states and transitions are distributed entities.

3) The extent of change caused by a transition is fixed; it does not depend on the state at which it occurs.

4) A transition is enabled to occur at a state iff the fixed extent of change of change associated with the transition is possible at that state.

BASIC LEVEL OF SYSTEM DESCRIPTION:

- atomic (local) states — conditions — B
- atomic (local) transitions — events — E

$B \cap E = \emptyset$: states and transitions are distinct entities

- distributed (global) state — case
  set of conditions holding concurrently
- distributed (global) transition — step
  set of events occurring concurrently
- transition relation
  specifies how cases are transformed into cases by the occurrences of steps

Key <u>questions</u>:

(1) When can a step occur (concurrently) at a case?

(2) What is the resulting case when a step occurs at a case?

~ · ~

The answers within net theory are given by postulating fixed neighbourhood relationship, flow relation, between the conditions and the events

⇓

"structural" transition relation relating <u>potential</u> cases to <u>potential</u> cases via <u>potential</u> steps.

~ · ~

by adding an initial case:

potential ⇝ actual

NETS

DEFINITION

A net is a triple $N = (S, T, F)$

(1) $S \cup T \neq \emptyset$ and $S \cap T = \emptyset$

(2) $F \subseteq (S \times T) \cup (T \times S)$

(3) $\underline{dom}(F) \cup \underline{ran}(F) = S \cup T$. □

~·~

$\underline{dom}(F) = \{x \in S \cup T : (x,y) \in F$ for some $y \in S \cup T\}$

$\underline{ran}(F) = \{y \in S \cup T : (x,y) \in F$ for some $x \in S \cup T\}$

~·~

S_N	S	S-elements (of N)
T_N	T	T-elements (of N)
X_N	$S \cup T$	elements (of N)
F_N	F	flow relation (of N)

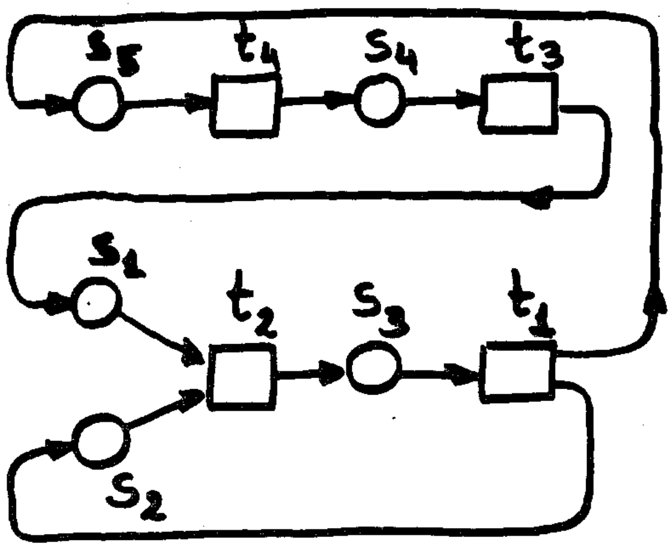
A net is an ordered bipartite directed graph without isolated nodes.

Hence there is a nice graphical notation to represent nets:

S-elements drawn as circles ○

T-elements drawn as boxes □

flow relation drawn as edges →



$N = (S, T, F)$

$S = \{s_1, \dots, s_5\}$, $T = \{t_1, \dots, t_4\}$

$F = \{(s_1, t_2), (s_2, t_2), (s_3, t_1),$
$(s_5, t_4), (s_4, t_3),$
$(t_2, s_3), (t_1, s_2), (t_1, s_5),$
$(t_4, s_4), (t_3, s_1)\}$

$N = (S, T, F)$

- $x \in X_N$, $Y \subseteq X_N$

$^{\bullet}x = \{y \in X_N : (y, x) \in F_N\}$

$x^{\bullet} = \{y \in X_N : (x, y) \in F_N\}$

$^{\bullet}Y = \bigcup_{x \in Y} {^{\bullet}x}$, $Y^{\bullet} = \bigcup_{x \in Y} x^{\bullet}$

~.~

$^{\bullet}t_2 = \{s_1, s_2\}$, $t_2^{\bullet} = \{s_3\}$,

$^{\bullet}t_3 = \{s_4\}$, $t_3^{\bullet} = \{s_1\}$

$^{\bullet}\{t_2, t_3\} = \{s_1, s_2, s_4\}$

$\{t_2, t_3\}^{\bullet} = \{s_1, s_3\}$

DEFINITION

Nets $N_1 = (S_1, T_1, F_1)$, $N_2 = (S_2, T_2, F_2)$ are *isomorphic*, $N_1 \equiv N_2$, iff $\exists$ bijections $\alpha : S_1 \to S_2$, $\beta : T_1 \to T_2$ such that

$\forall p \in S_1 \;\; \forall t \in T_1$

$(p, t) \in F_1$ iff $(\alpha(p), \beta(t)) \in F_2$

$(t, p) \in F_1$ iff $(\beta(t), \alpha(p)) \in F_2$

$\forall t \in T_1$

$$\underbrace{\alpha({}^\bullet t)}_{N_1} = \underbrace{{}^\bullet\beta(t)}_{N_2} \;\&\; \underbrace{\alpha(t^\bullet)}_{N_1} = \underbrace{\beta(t)^\bullet}_{N_2}$$

DEFINITION

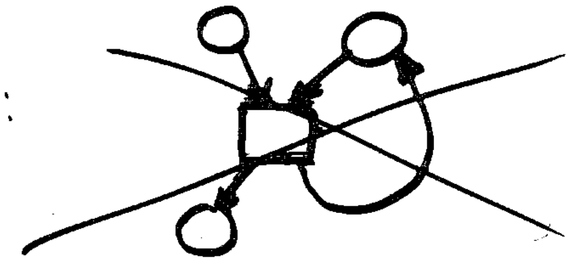
A net N is:

- pure iff $(\forall x)_{X_N} \; [{}^\bullet x \cap x^\bullet = \emptyset]$
- simple iff $(\forall x, y)_{X_N}$

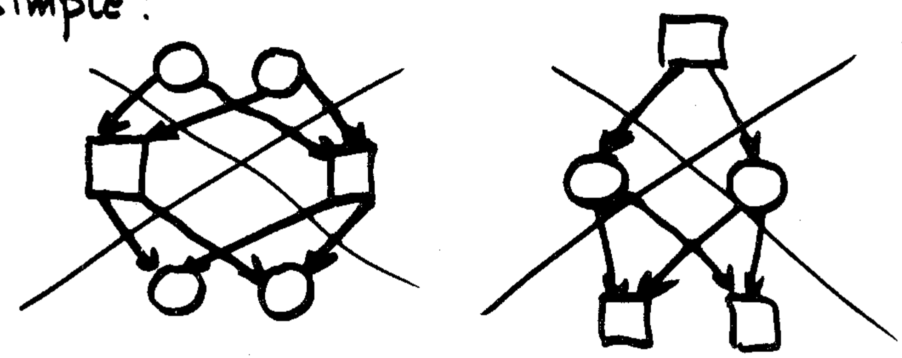
$$[({}^\bullet x = {}^\bullet y \;\&\; x^\bullet = y^\bullet) \implies (x = y)]$$

~ · ~

pure:



simple:



Depending on applications various interpretations can be given to the elements of a net.

We will use a net to represent the (static) underlying structure of a distributed system.

conditions (local states) represented by S-elements

events (local transitions) represented by T-elements

neighbourhood relationship represented by F

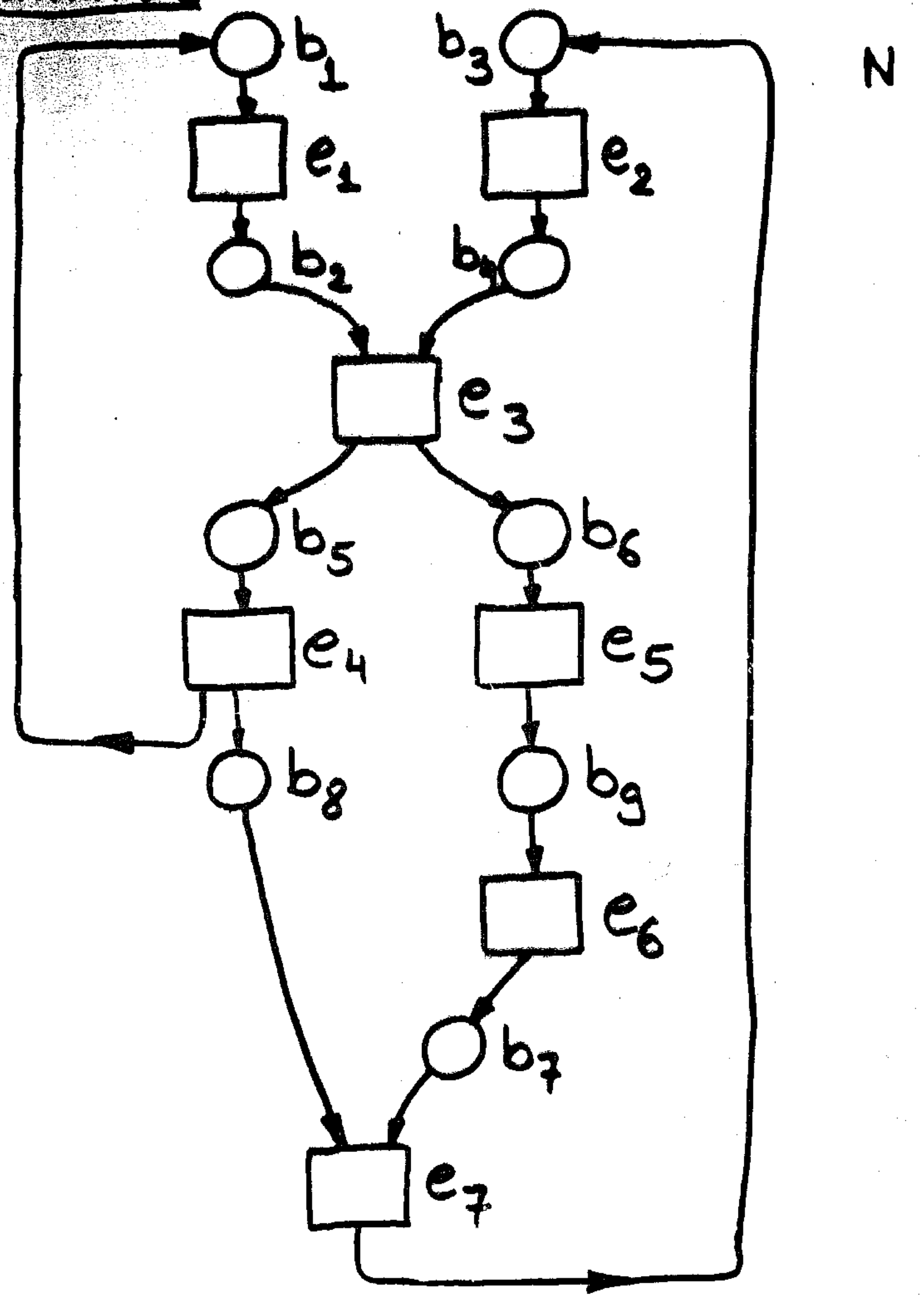
Accordingly:

$N = (B, E, F)$

$e \in E$: ${}^{\bullet}e$ pre-conditions of e

$e^{\bullet}$ post-conditions of e

EXAMPLE



$N=(B,E,F)$, $C\subseteq B$, $e\in E$

Q: When can e occur at C ?

A: e can occur at C iff
all pre-conditions hold at C ($^\bullet e\subseteq C$)
& no post-conditions hold at C ($e^\bullet\cap C=\emptyset$)

$C\,[e\rangle_N$

$\{b_1\}\,[e_1\rangle_N$

$\{b_1,b_6\}\,[e_1\rangle_N$

$\{b_2,b_4,b_8\}\,[e_3\rangle_N$

$\neg\,\{b_1,b_6\}\,[e_3\rangle_N$

$\neg\,\{b_1,b_2\}\,[e_1\rangle_N$

Note that if $C\,[e\rangle_N$

then $^\bullet e\cap e^\bullet=\emptyset$

Hence we often consider only pure nets (no loops)

$N = (B, E, F)$ $C \subseteq B$ $e \in E$

e can occur at C

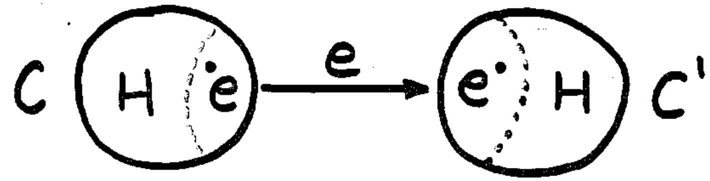
Q: What is the result of e occurring at C?

A: When e occurs at C, the pre-conditions of e cease to hold and the post-conditions of e begin to hold; the remaining part of the case remains unaffected (hence the resulting case C' is $(C - {}^{\bullet}e) \cup e^{\bullet}$). $C[e\rangle_N C'$

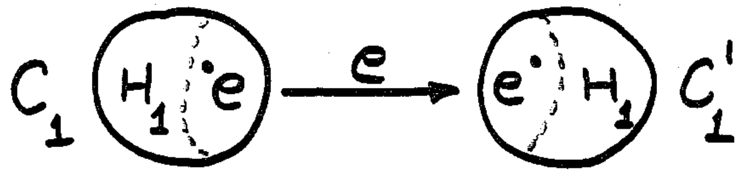
Hence the change-of-state produced by an event occurrence is confined strictly to its immediate neighbourhood

$N = (B, E, F)$, $C, C' \subseteq B$, $e \in E$

$C[e\rangle_N C'$ iff $C - C' = {}^{\bullet}e$ & $C' - C = e^{\bullet}$



C' uniquely determ. by C and e



Hence the change caused by e does not depend on a global state in which it occurs!!!

$$\{b_1\} [e_1>_N \{b_2\}$$

$$\{b_1, b_6\} [e_1>_N \{b_2, b_6\}$$

$$\{b_2, b_4, b_8\} [e_3>_N \{b_5, b_6, b_8\}$$

$N = (B, E, F)$, $C \subseteq B$, $U \subseteq E$

Q: When can the events in U occur concurrently at C?
(When can the step U occur at C?)

A: U can occur at C iff the events in U can individually occur at C without interfering with each other. $C [U>_N$

Since the effect of an occurrence of the event e is confined to ${}^\bullet e \cup e^\bullet$ the "non-interfering" can be formalized as:

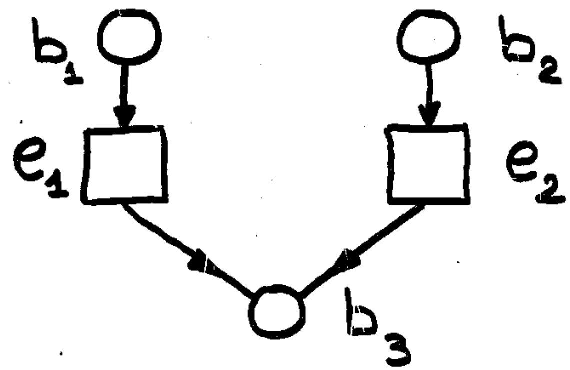
$$(\forall e_1, e_2)_U \left[(e_1 \neq e_2) \Longrightarrow ({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \emptyset \right]$$

$\{b_1, b_3, b_6\} [\{e_1, e_2\} >_N$

$\{b_2, b_4, b_9\} [\{e_3, e_6\} >_N$

$\neg \{b_1, b_3, b_6\} [\{e_6, e_2\} >_N$

because $\neg \{b_1, b_3, b_6\} [e_6 >_N$



$\neg \{b_1, b_2\} [\{e_1, e_2\} >_N$

because $({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet)$

$= \{b_3\} \neq \emptyset$

e_1 and e_5 interfere with each other

$N = (B, E, F)$, $C \subseteq B$, $\mathcal{U} \subseteq T$

$\mathcal{U}$ is enabled to occur at C

Q: What is the result of $\mathcal{U}$ occuring at C ?

A: The result is the "sum" of the results of the events in $\mathcal{U}$ occuring individually at C (hence when $\mathcal{U}$ occurs at C the resulting case C' is given by: $C' = (C - {}^\bullet\mathcal{U}) \cup \mathcal{U}^\bullet$)

$$C\,[\mathcal{U}\rangle_N\, C'$$

$$\{b_1, b_3, b_6\}\,[\{e_1, e_2\}\rangle_N\, \{b_2, b_4, b_6\}$$

$$\{b_2, b_4, b_9\}\,[\{e_3, e_6\}\rangle_N\, \{b_5, b_6, b_7\}$$

DEFINITION

Let N be a net and let $\mathcal{U} \subseteq E_N$.

(1) $\mathcal{U}$ is independent, $\underline{ind}_N(\mathcal{U})$, iff $(\forall e_1, e_2)_{\mathcal{U}}$ [if $e_1 \neq e_2$ then $({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \emptyset$].

(2) Let $C \subseteq B_N$.
$\mathcal{U}$ is a step enabled at C, $C\,[\mathcal{U}\rangle_N$, iff $\underline{ind}_N(\mathcal{U})$, ${}^\bullet\mathcal{U} \subseteq C$ and $\mathcal{U}^\bullet \cap C = \emptyset$.

(3) Let $C_1, C_2 \subseteq B_N$.
$\mathcal{U}$ is a step leading from C_1 to C_2, $C_1\,[\mathcal{U}\rangle_N\, C_2$, iff $C_1\,[\mathcal{U}\rangle_N$ and $C_2 = (C_1 - {}^\bullet\mathcal{U}) \cup \mathcal{U}^\bullet$.

THEOREM

$N = (B, E, F)$, $C, D \subseteq B$, $U \subseteq T$.

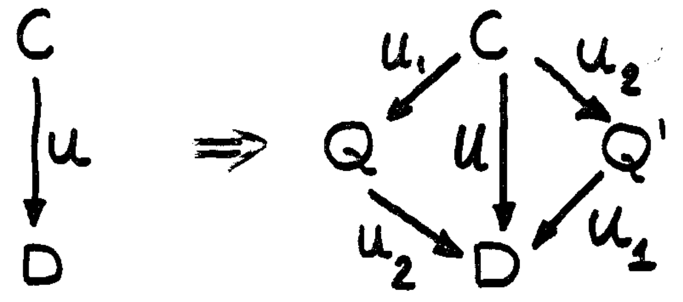
Let $\{U_1, U_2\}$ be a partition of U

$(U_1, U_2 \neq \emptyset,\ U_1 \cap U_2 = \emptyset,\ U_1 \cup U_2 = U)$

If $C[U>_N D$, then $\exists\, Q \subseteq B$ such that

$C[U_1>_N Q$ and $Q[U_2>_N D$

—



diamond property

DEFINITION

Let N be a net and let $C \subseteq B_N$.

The forward case class generated by C, $[C>_N$, is the smallest subset of 2^{B_N} such that:

1) $C \in [C>_N$,

2) if $C_1 \in [C>_N$, and $C_2 \subseteq B_N$ is such that $C_1 [U>_N C_2$ for some $U \subseteq E_N$,

then $C_2 \in [C>_N$.

$N = (B, E, F)$, $C \subseteq B$,

$$\tau = e_1 e_2 \ldots e_n \in E^+, \; n \geq 1$$

Q: When can τ occur at C?

A: τ can occur at C iff the events in τ can individually occur in the order determined by τ $\qquad C[\tau\rangle_N$

$\exists\, C_0, C_1, \ldots, C_n \subseteq B$ such that

$C_0 = C$ and

$\forall i \in \{1, \ldots, n\} \quad C_{i-1}[e_i\rangle_N C_i$

we write:

$$C_0[e_1\rangle_N C_1[e_2\rangle_N C_2 \ldots C_{n-1}[e_n\rangle C_n$$

Note that

$C_1 C_2 \ldots C_n$ uniquely determined by C and τ

because

$$C_0 - C_1 = {}^\bullet e_1 \;,\quad C_1 - C_0 = e_1^\bullet$$

$$C_1 - C_2 = {}^\bullet e_2 \;,\quad C_2 - C_1 = e_2^\bullet$$

$$\vdots \qquad\qquad \vdots$$

$$C_{n-1} - C_n = {}^\bullet e_n \;,\quad C_n - C_{n-1} = e_n^\bullet$$

$N = (B, E, F)$, $C \subseteq B$,

$\tau = e_1 e_2 \ldots e_n$ can occur at C

Q: What is the result of τ occuring at C ?

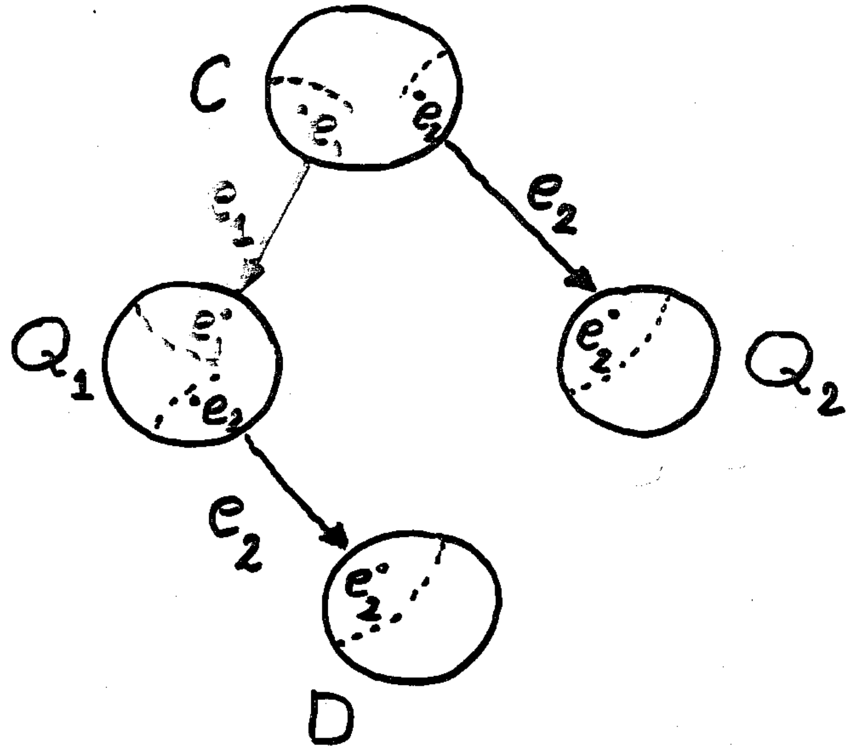
A: The unique $C_n \subseteq B$ such that

$$C = C_0 [e_1\rangle_N C_1 \ldots . C_{n-1} [e_n\rangle_N C_n$$

$N = (B, E, F)$, $C \subseteq B$, $e_1, e_2 \in E$

If $C [e_1 e_2\rangle_N$ and $C [e_2\rangle_N$,

then $C [\{e_1, e_2\}\rangle_N$.



$$\underbrace{{}^\bullet e_1, {}^\bullet e_2 \subseteq C,\ e_1^\bullet \cap C = \emptyset,\ e_2^\bullet \cap C = \emptyset}$$

$$\Downarrow$$

(1) $({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \emptyset$ iff ${}^\bullet e_1 \cap {}^\bullet e_2 = \emptyset$ & $e_1^\bullet \cap e_2^\bullet = \emptyset$

—

$$\underbrace{{}^\bullet e_1 \cap Q_1 = \emptyset \ \&\ {}^\bullet e_2 \subseteq Q_1}$$

$$\Downarrow$$

(2) ${}^\bullet e_1 \cap {}^\bullet e_2 = \emptyset$

—

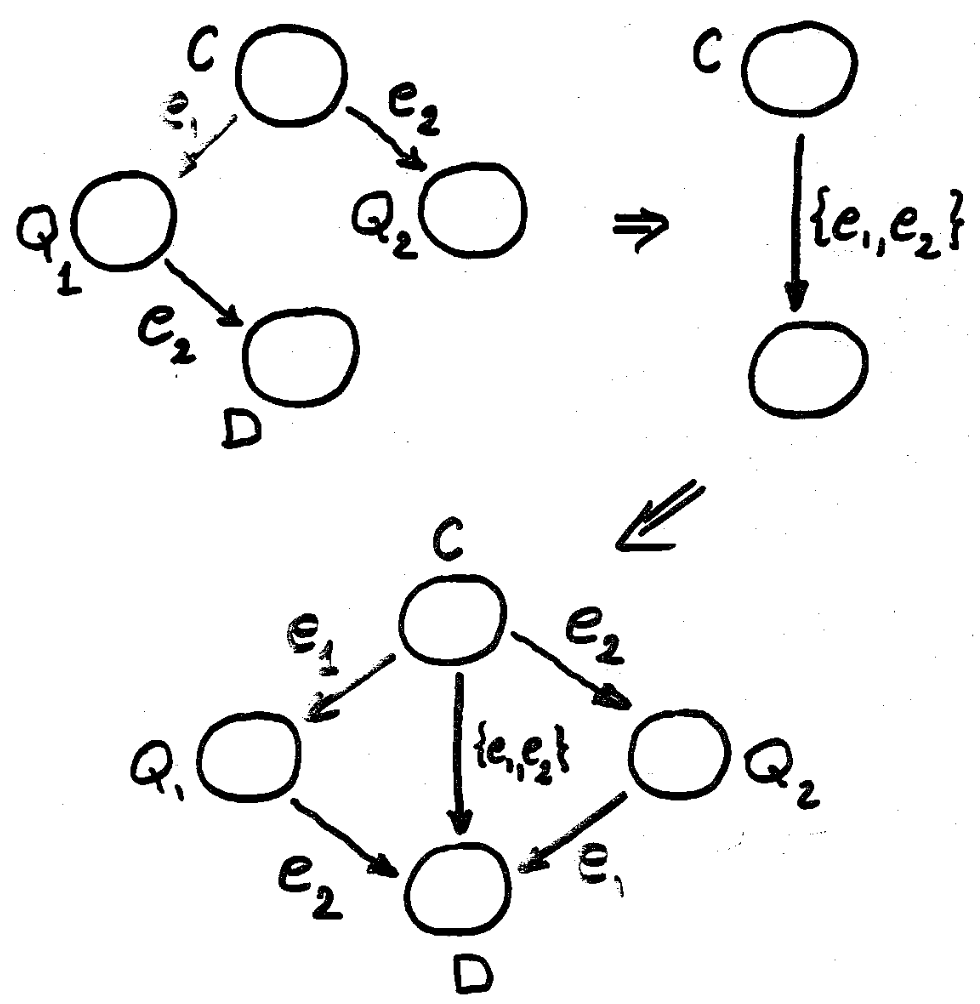
$$\underbrace{e_1^\bullet \subseteq Q_1 \ \&\ e_2^\bullet \cap Q_1 = \emptyset}$$

$$\Downarrow$$

(3) $e_1^\bullet \cap e_2^\bullet = \emptyset$

(1) & (2) & (3) $\Longrightarrow$ $\underline{ind}_N(\{e_1, e_2\})$ (4)

$C[e_1>_N$ & $C[e_2>_N$ & (4) $\Rightarrow C[\{e_1,e_2\}>$..



THEOREM

$N = (B, E, F)$, $C, D \subseteq B$,

$\emptyset \neq \mathcal{U} \subseteq E$.

(1) $C[\mathcal{U}>_N$ iff

$\forall$ ordering $e_1, \ldots, e_n$ of $\mathcal{U}$

$C[e_1 \ldots e_n>_N$.

(2) $C[\mathcal{U}>_N D$ iff

$\forall$ ordering $e_1, \ldots, e_n$ of $\mathcal{U}$

$C[e_1 \ldots e_n>_N D$.

sequentialization property

ELEMENTARY NET SYSTEMS

DEFINITION

An elementary net system (EN system for short) is a 4-tuple $\mathcal{N} = (B, E, F, C_{in})$ where

(B, E, F) is a net called the underlying net of $\mathcal{N}$, <u>und</u>(N),

$C_{in} \subseteq B$ is the initial case of $\mathcal{N}$, <u>inc</u>$(\mathcal{N})$.

~ . ~

We carry over to EN systems the notation and the terminology concerning nets

$B_{\mathcal{N}}$

$E_{\mathcal{N}}$

$F_{\mathcal{N}}$

EN system as an abstract model of a distributed system :

$$\mathcal{N} = (\underbrace{B, E, F}, \underbrace{C_{in}})$$

(B, E, F) → underlying static structure

C_{in} → dynamic behaviour (actual state space)

$\mathcal{C}_{\mathcal{N}} = [C_{in}\rangle_N$ the set of cases of $\mathcal{N}$

$$U_{\mathcal{N}} = \{ U \subseteq E : (\exists C_1, C_2)_{\mathcal{C}_{\mathcal{N}}} [C_1 [U\rangle_{\mathcal{N}} C_2] \}$$

the set of steps of $\mathcal{N}$

Graphical notation for an EN system consists of
the graphical notation for the underlying net, and
the marking of C_{in} by tokens

Playing the token game we can "compute" $\mathcal{C}_N$ and $\mathcal{U}_N$.

FUNDAMENTAL SITUATIONS

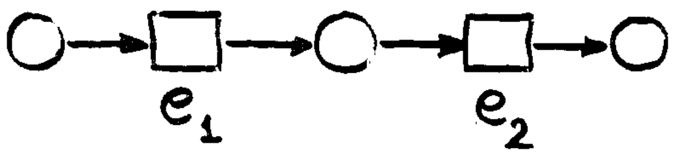
Let $\mathcal{N}$ be an EN system, let $C \in \mathcal{C}_{\mathcal{N}}$, and let $e_1, e_2 \in E_{\mathcal{N}}$.

e_1, e_2 can be related to each other at C in (at least) three ways.

Sequence

e_1 can occur at C but not e_2. However, after e_1 has occurred e_2 can occur.

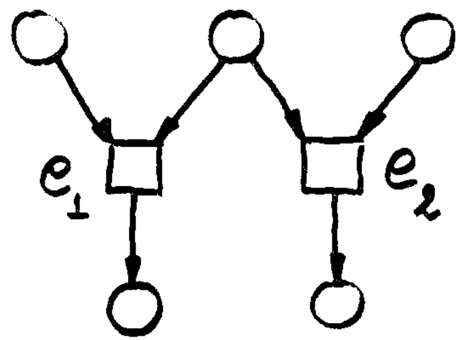
e_1, e_2 are in sequence at C iff $C[e_1>$, $\neg(C[e_2>)$, and $C'[e_2>$, where $C[e_1>C'$.



Choice (conflict)

e_1 and e_2 can occur individually at C but they cannot occur together at C: $\{e_1, e_2\}$ is not a step at C. (Whether e_1 or e_2 will occur at C is left unspecified: in this way $\mathcal{N}$ exhibits nondeterminism).

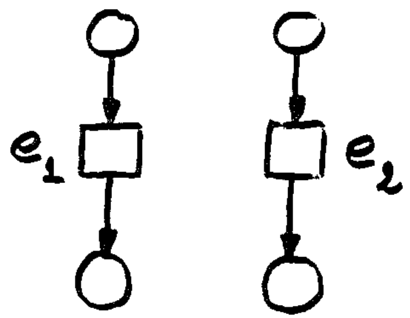
e_1, e_2 are in conflict at C iff $C[e_1>$, $C[e_2>$, and $\neg(C[\{e_1, e_2\}>)$.



Concurrency

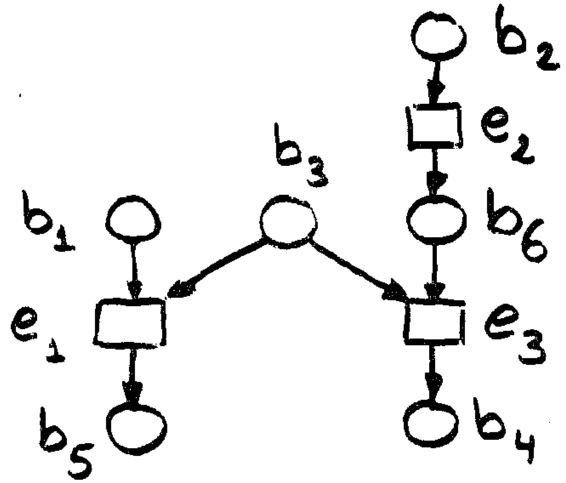
e_1 and e_2 can occur at C without interfering with each other. Moreover no order is specified over their occurrences. Hence, in general, the occurrences of events and the resulting holdings of conditions will be partially ordered: in this way $\mathcal{N}$ exhibits non-sequential behaviour.

e_1, e_2 can occur concurrently at C iff $C[\{e_1, e_2\}\rangle$



A mixture of concurrency and conflict may result in a situation called confusion.

Confusion



$\{b_1, b_2, b_3\}[\{e_1, e_2\}\rangle\{b_5, b_6\}$

σ_1: first e_1 with <u>no conflict</u>; then e_2

σ_2: first e_2, then <u>conflict</u> between e_1, e_3: the conflict resolved in favour of e_1 which then occurred.

Let $\mathcal{N}$ be an EN system, let $C \in \mathcal{C}_{\mathcal{N}}$

- Let $e \in E$ be such that $C[e\rangle$.

The conflict set of e (at C), $\underline{cfl}(e, C)$, is the set

$\{e' \in E : C[e'\rangle \text{ and } \neg(C[\{e, e'\}\rangle)\}$.

- For $e_1, e_2 \in E$ such that $C[\{e_1, e_2\}\rangle$, the triplet (C, e_1, e_2) is a confusion (at C) iff $\underline{cfl}(e_1, C) \neq \underline{cfl}(e_1, C_2)$, where $C[e_2\rangle C_2$.
- $\mathcal{N}$ is confused at C iff there is a confusion at C.

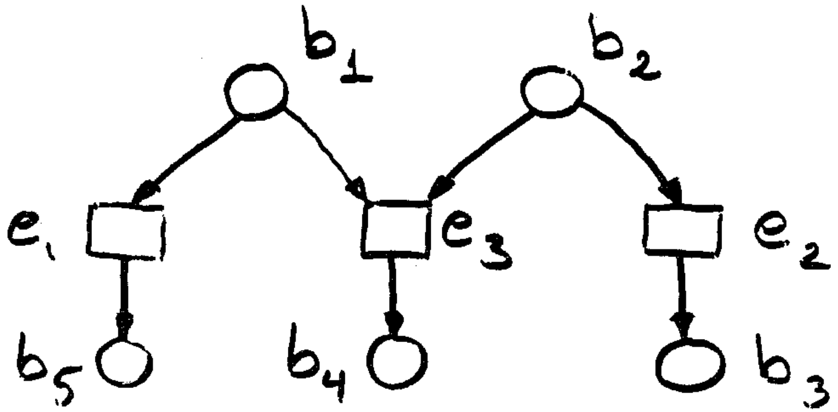
$C = \{b_1, b_2, b_3\}$.

$\underline{cfl}(e_1, C) = \emptyset$

(C, e_1, e_2) is a confusion

$\underline{cfl}(e_1, C) = \emptyset \neq \{e_3\} = \underline{cfl}(e_1, C_2)$

$C_2 = \{b_1, b_3, b_4\}$



$C = \{b_1, b_2\}$

(C, e_1, e_2) is a confusion

$\underline{cfl}(e_1, C) = \{e_3\} \neq \emptyset = \underline{cfl}(e_1, C_2)$

$C_2 = \{b_1, b_3\}$

One may have then

conflict increasing confusions

conflict decreasing confusions,

and

confusions that are neither

conflict increasing nor conflict

decreasing.

STATE SPACES OF EN SYSTEMS

Formalizing the state space

DEFINITION

Let $\mathcal{N}$ be an EN system.

(1) The case graph of $\mathcal{N}$, $CG(\mathcal{N})$, is the initialized edge-labeled graph $((V, Y), v_{in})$ such that $V = \mathcal{C}_{\mathcal{N}}$, $v_{in} = \underline{inc}(\mathcal{N})$, and

$$Y = \{(C_1, U, C_2) : C_1, C_2 \in \mathcal{C}_{\mathcal{N}},\ U \in \mathcal{U}_{\mathcal{N}}, \text{ and } C_1 [U \rangle_{\mathcal{N}} C_2\}$$

(2) The sequential case graph of $\mathcal{N}$, $SCG(\mathcal{N})$, is the initialized edge-labeled graph $((V, Y), v_{in})$ such that $V = \mathcal{C}_{\mathcal{N}}$, $v_{in} = \underline{inc}(\mathcal{N})$, and

$$Y = \{(C_1, \{e\}, C_2) : C_1, C_2 \in \mathcal{C}_{\mathcal{N}},\ e \in E, \text{ and } C_1 [\{e\} \rangle_{\mathcal{N}} C_2\}.$$

$N=(B,E,F)$, $C\subseteq B$

Let $FS_N(C)=\{\tau\in E^*: C[\tau>_N\}$

Then

$[C>_N=\{D\subseteq B: \exists\tau\in FS_N(C)$ such that $C[\tau>_N D\}$.

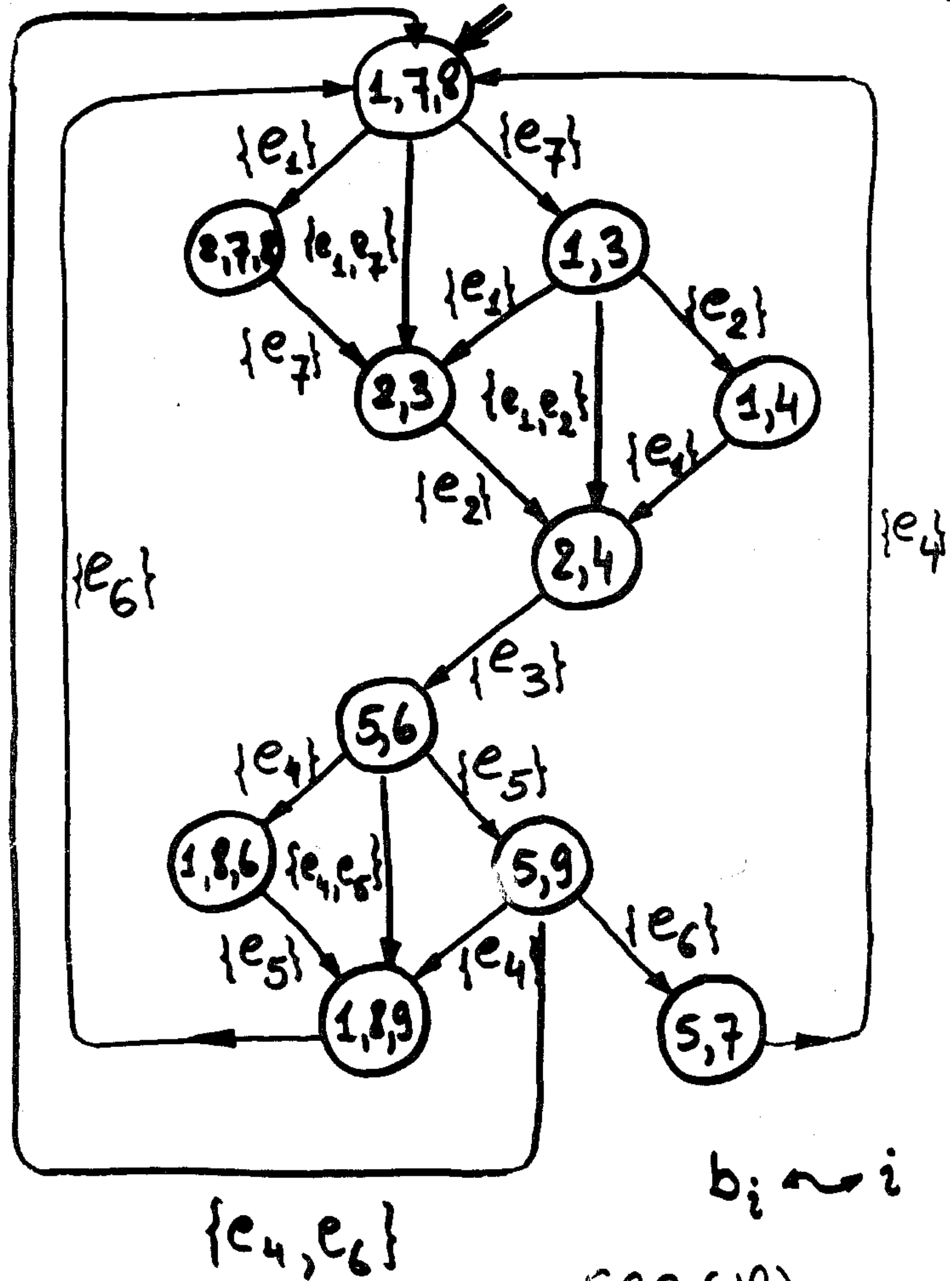
In particular, for $\mathcal{N}=(B,E,F,C_{in})$

$\mathcal{C}_{\mathcal{N}}=[C_{in}>_{\mathcal{N}}=$

$\{D\subseteq B: \exists\tau\in FS_{\mathcal{N}}(C_{in})$ such that $C_{in}[\tau>_{\mathcal{N}} D\}$.

Thus $SCG(\mathcal{N})$ is strongly connected (connected from v_{in}).

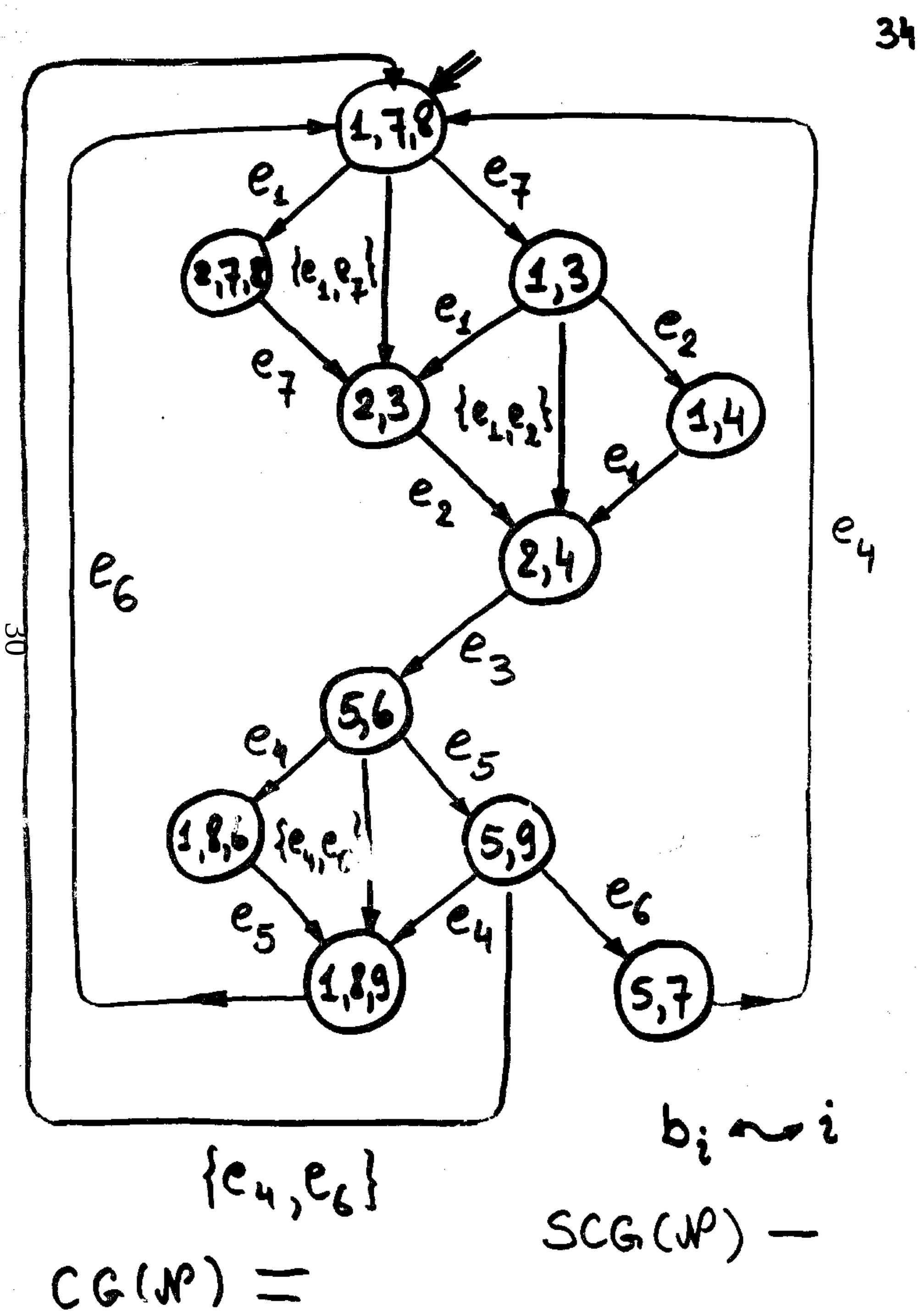
The difference between $SCG(\mathcal{N})$ and $CG(\mathcal{N})$ is in labelled edges only.

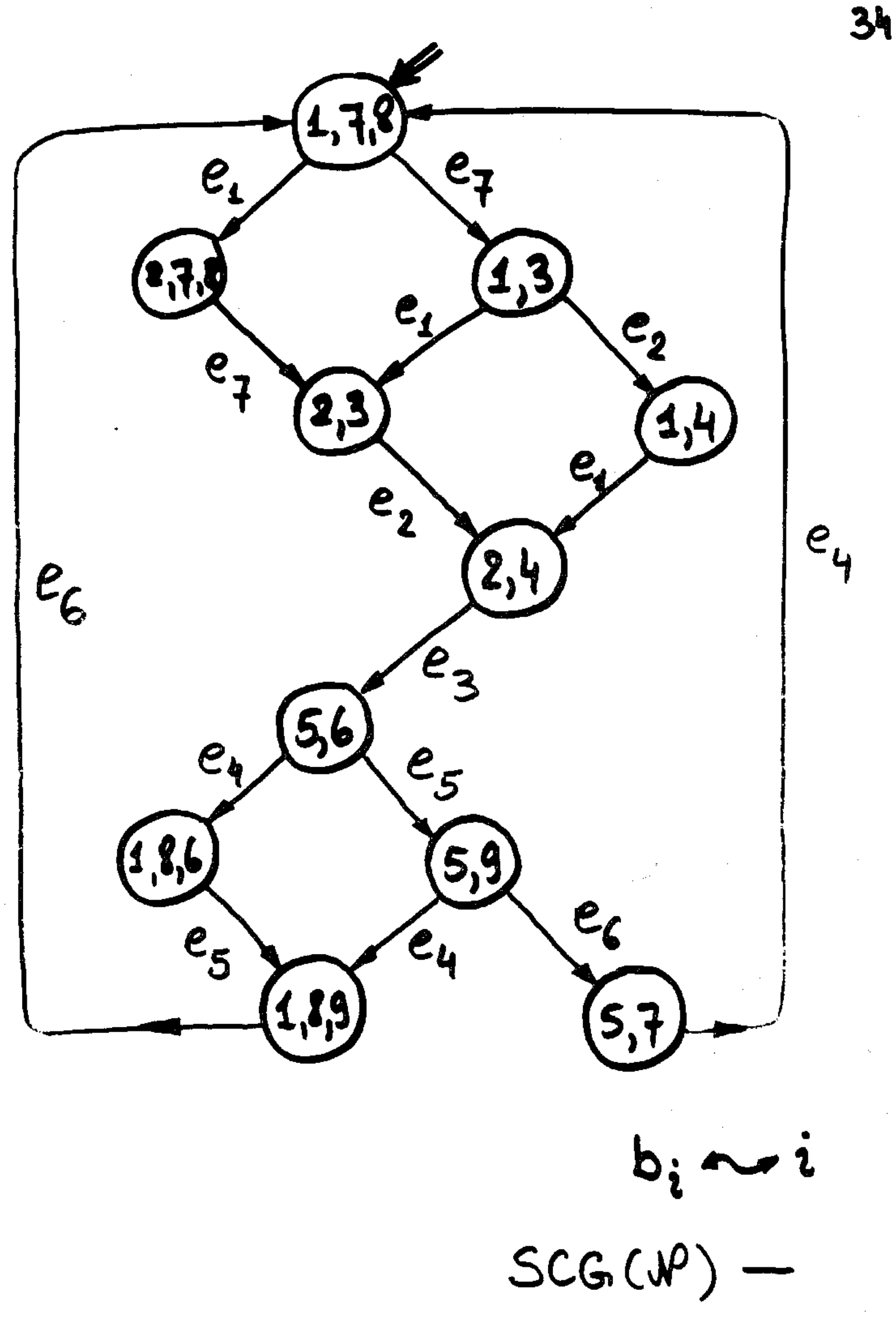


$b_i \rightsquigarrow i$

$SCG(\mathcal{N})$ —

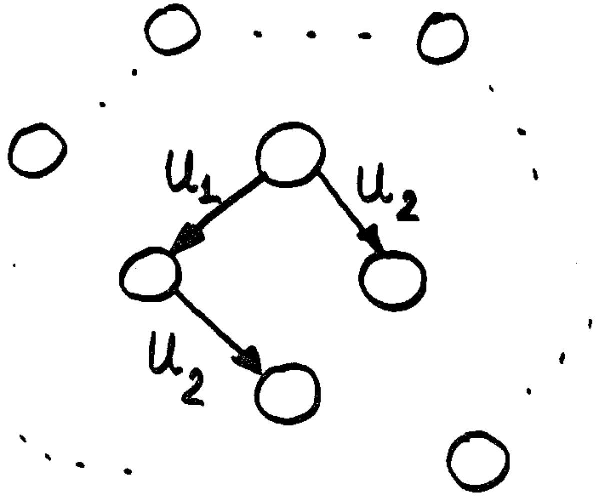
$CG(\mathcal{N})$ =


34
1,7,8
2,7,8
1,3
2,3
1,4
2,4
5,6
1,8,6
5,9
1,8,9
5,7
e1
e7
{e1,e7}
e1
e2
e7
{e1,e2}
e1
e2
e3
e4
e5
e4
e5
e6
e6
e4
{e4,e6}
b_i ⤳ i
SCG(N) —
CG(N) =

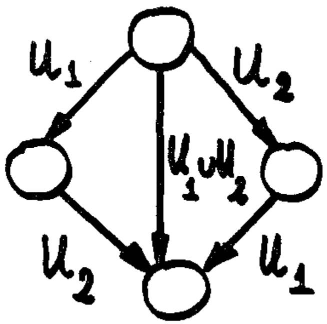

34
1,7,8
2,7,8
1,3
2,3
1,4
2,4
5,6
1,8,6
5,9
1,8,9
5,7
e1
e7
e1
e2
e7
e1
e2
e3
e4
e5
e5
e4
e6
e6
e4
b_i ⤳ i
SCG(N) —

◇-rule (for edge-labeled graphs where labels are sets):

if



then



The ◇-closure for a graph g, $\diamond^*(g)$, is the graph h resulting from g by applying the ◇-rule as long as possible.

THEOREM

$$(\forall \mathcal{N})_{EN} \left[\diamond^*(SCG(\mathcal{N})) = CG(\mathcal{N}) \right] \square$$

The case graph of $\mathcal{N}$ can be "syntactically" recovered from the sequential case graph of $\mathcal{N}$!!!

g_1, g_1' initialized edge-labeled graphs

$g_1 = ((V_1, E_1), v_1)$ $\underline{alph}(g_1) = \Delta_1$

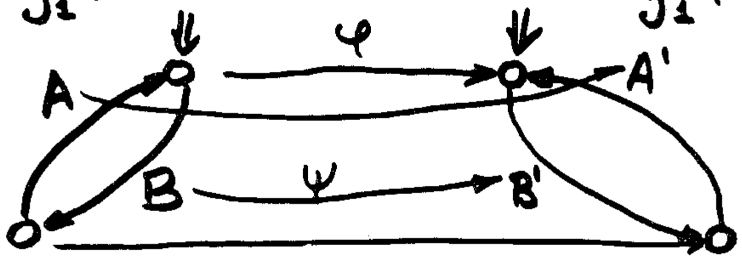
$g_1' = ((V_1', E_1'), v_1')$ $\underline{alph}(g_1') = \Delta_1'$

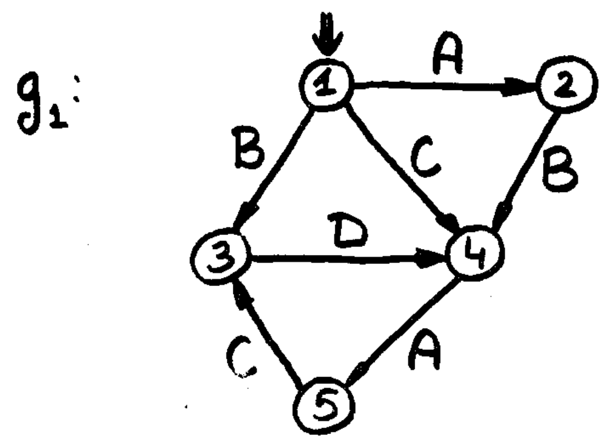
An isomorphism from g_1 onto g_1' is a pair of bijections $\varphi : V_1 \to V_1'$, and $\psi : \Delta_1 \to \Delta_1'$ such that

- $\varphi(v_1) = v_1'$,
- $(u_1, A, u_2) \in E_1$ iff $(\varphi(u_1), \psi(A), \varphi(u_2)) \in E_2$.

g_1 $\underline{lisom}$ g_2

~·~



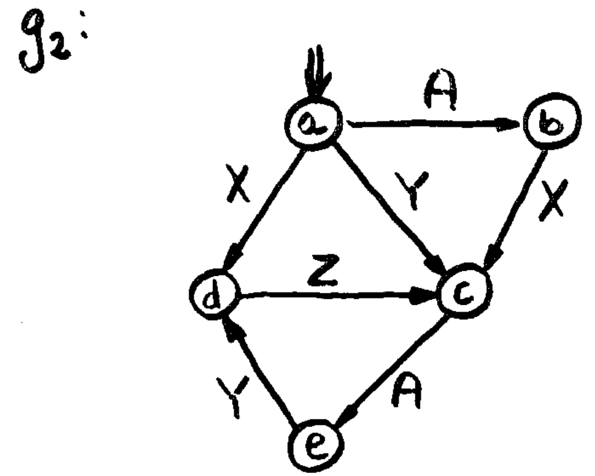


φ:

1	2	3	4	5
a	b	d	c	e

ψ:

A	B	C	D
A	X	Y	Z



g_1 $\underline{lisom}$ g_2

DEFINITION

EN systems $\mathcal{N}_1, \mathcal{N}_2$ are state space similar, $\mathcal{N}_1 \cong \mathcal{N}_2$, iff $CG(\mathcal{N}_1)$ lisom $CG(\mathcal{N}_2)$.

THEOREM

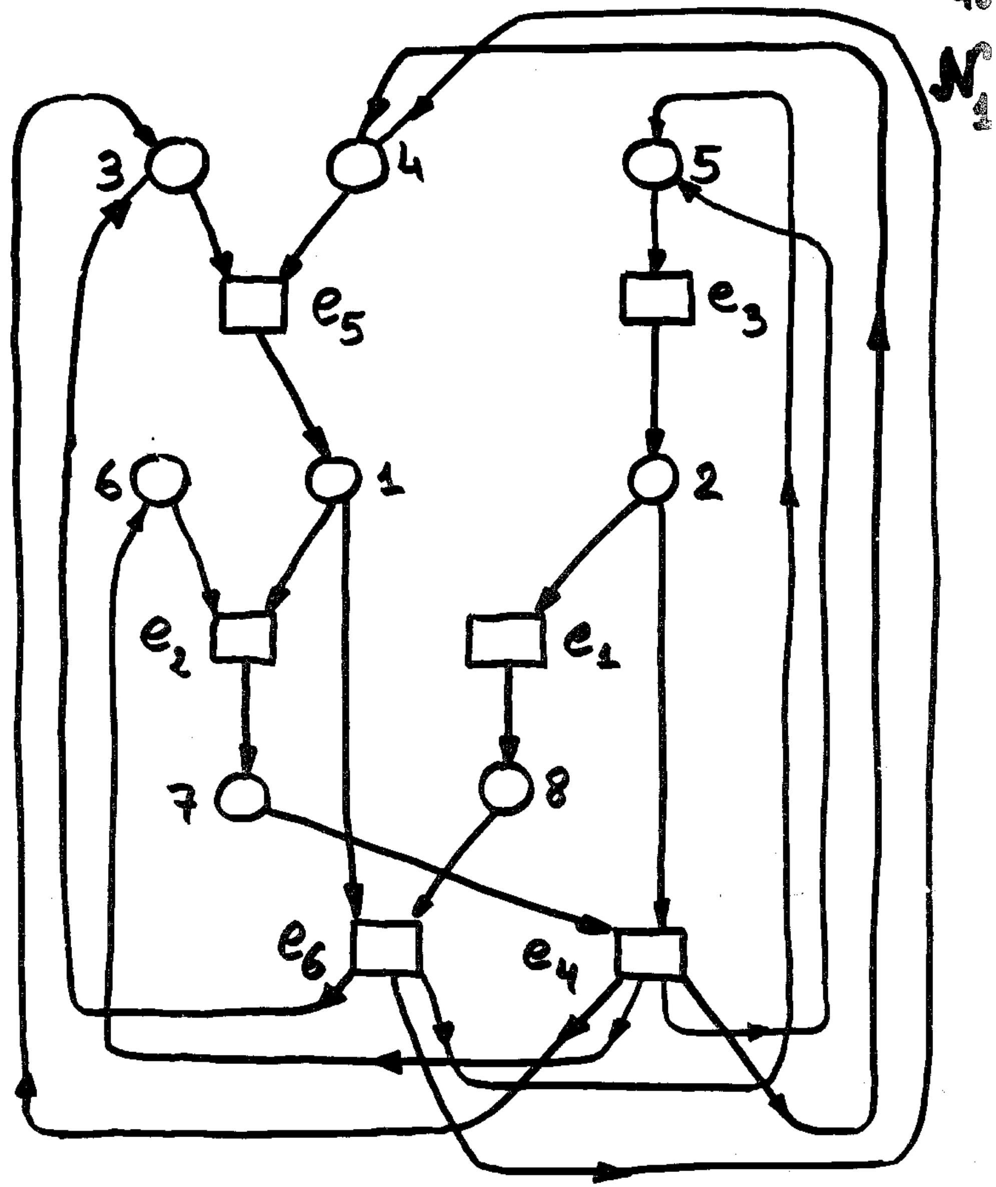
Let $\mathcal{N}_1, \mathcal{N}_2$ be EN systems.

1) $CG(\mathcal{N}_1)$ lisom $CG(\mathcal{N}_2)$

iff

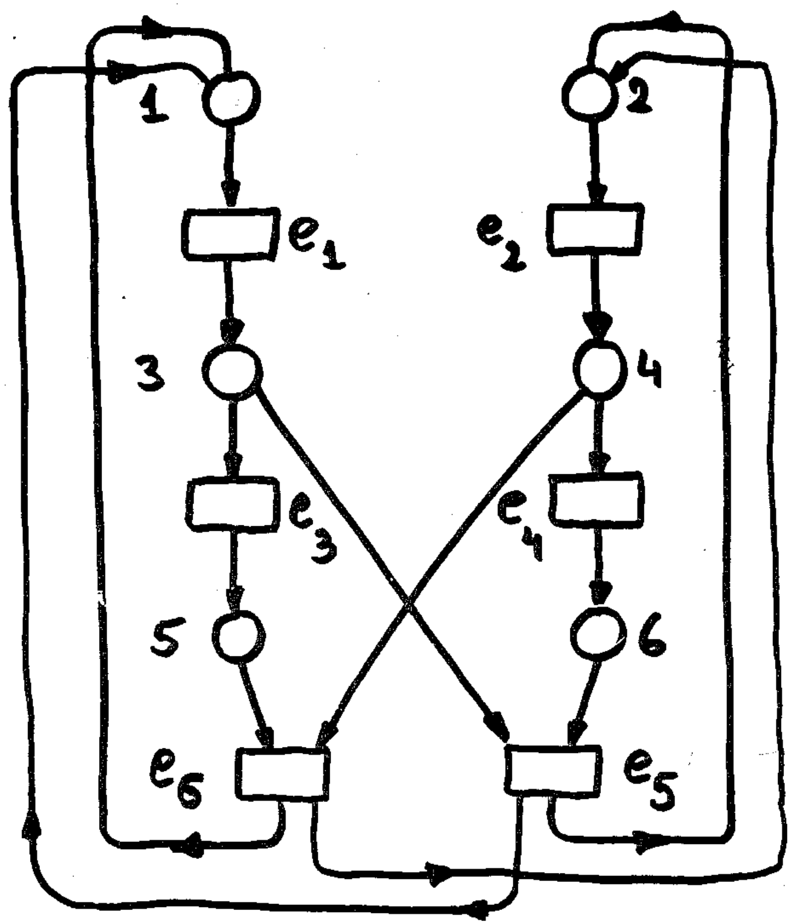
$SCG(\mathcal{N}_1)$ lisom $SCG(\mathcal{N}_2)$.

2) $\mathcal{N}_1 \cong \mathcal{N}_2$ iff $SCG(\mathcal{N}_1)$ lisom $SCG(\mathcal{N}_2)$.

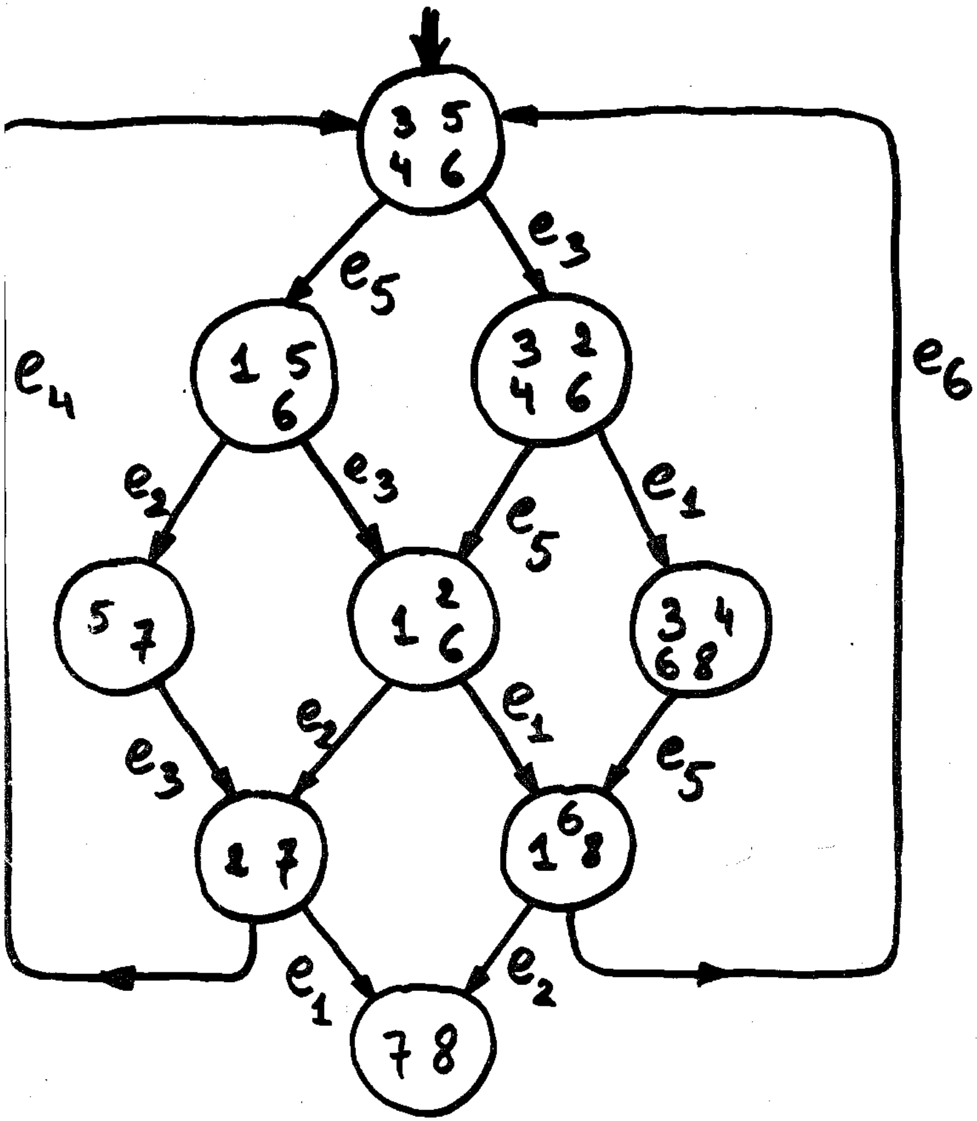


inc $(\mathcal{N}_1) = \{3, 4, 5, 6\}$

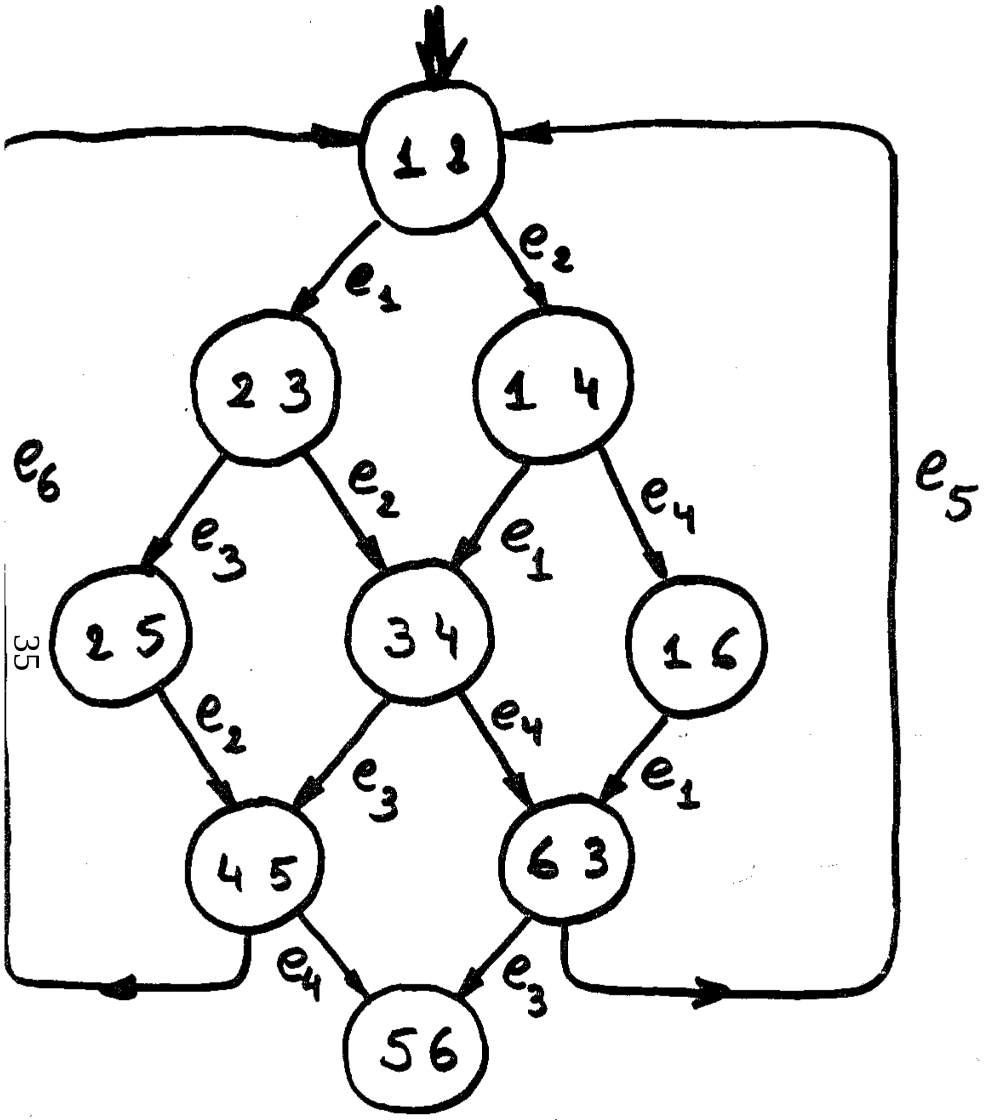
$\mathcal{N}_2$



$\underline{ine}(\mathcal{N}_2) = \{1, 2\}$



$SCG(\mathcal{N}_1)$



SCG($\mathcal{N}_2$)

$$\mathcal{N}_1 \cong \mathcal{N}_2$$

$\varphi(\{3,4,5,6\}) = \{1,2\}$

$\varphi(\{1,5,6\}) = \{2,3\}$

$\varphi(\{2,3,4,6\}) = \{1,4\}$

$\varphi(\{5,7\}) = \{2,5\}$

$\varphi(\{1,2,6\}) = \{3,4\}$

$\varphi(\{3,4,6,8\}) = \{1,6\}$

$\varphi(\{2,7\}) = \{4,5\}$

$\varphi(\{1,6,8\}) = \{3,6\}$

$\varphi(\{7,8\}) = \{5,6\}$

ψ:

e_1	e_2	e_3	e_4	e_5	e_6
e_4	e_3	e_2	e_6	e_1	e_5

OFTEN MADE (DESIRED) ASSUMPTIONS

SIMPLE EN SYSTEMS

An EN system $\mathcal{N}$ is simple iff $N = \underline{\mathbf{und}}(\mathcal{N})$ is simple

$$((\forall x, y \in X_N)\,[\,{}^\bullet x = {}^\bullet y \;\&\; x^\bullet = y^\bullet \text{ implies } x = y\,])$$

~·~

In an EN system $\mathcal{N}$ the change caused by an event occurrence is the same in every context:

$$(\forall e)_E\,[\text{if } (C_1, e, C_2), (C_3, e, C_4) \in CG(\mathcal{N})$$

$$\text{then } (C_1 - C_2,\, C_2 - C_1) = (C_3 - C_4,\, C_4 - C_3)]$$

On the other hand we may have

and different events e_1, e_2 with $^{\bullet}e_1 = {^{\bullet}e_2} = X$ & $e_1^{\bullet} = e_2^{\bullet} = Y$.

This cannot happen in an (event) simple EN system.

THEOREM

$\mathcal{N} = (B, E, F, C_{in})$ <u>simple</u> EN syst

$\forall e_1, e_2 \in B \quad \forall C_1, D_1, C_2, D_2 \subseteq B$

If $C_1[e_1 > D_1$ & $C_2[e_2 > D_2$,

then $e_1 = e_2$ iff

$C_1 - D_1 = C_2 - D_2$ & $D_1 - C_1 = D_2 - C_2$.

extensionality principle

Note that

$C_1 - D_1 = {^{\bullet}e_1} \qquad D_1 - C_1 = e_1^{\bullet}$

$C_2 - D_2 = {^{\bullet}e_2} \qquad D_2 - C_2 = e_2^{\bullet}$

Thus

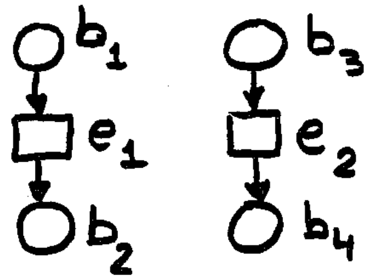
$e_1 = e_2$ iff $({^{\bullet}e_1}, e_1^{\bullet}) = ({^{\bullet}e_2}, e_2^{\bullet})$

For an event e

$({^{\bullet}e}, e^{\bullet})$ characteristic pair of e

REDUCED EN SYSTEMS

$\mathcal{N}$:



$C_{in} = \{b_3\}$

e_1 useless e_2 useful

$e \in E$ is useful iff
$\exists C \in \mathcal{C}_{\mathcal{N}}$ $C[e\rangle$,
otherwise e is useless.

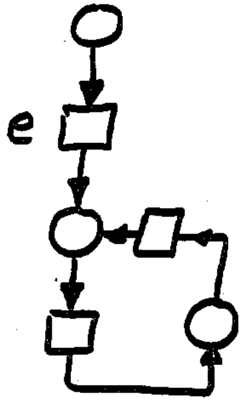
$\mathcal{N}$ is reduced iff
$\forall e \in E$ e is useful.

Note that e is useful iff
e is an edge-label in $SCG(\mathcal{N})$

"Mostly" it is assumed that an EN system is reduced.

THEOREM
$(\forall \text{EN } \mathcal{N})(\exists \text{EN } \mathcal{N}')$
$[\mathcal{N}'$ is reduced & $\mathcal{N} \cong \mathcal{N}']$.

Various degrees of usefulness



e is useful

but not live

$e \in E$ is live iff

$$\forall C \in \mathcal{C}_{\mathcal{N}}\ \exists \tau \in E^{*}\ \exists D \in \mathcal{C}_{\mathcal{N}}$$

$$C[\tau > D \ \& \ D[e > .$$

CONTACT-FREE EN SYSTEMS

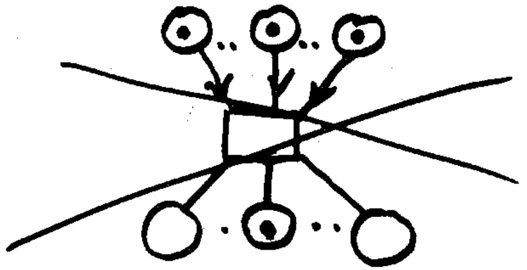
In an EN system $\mathcal{N}$ an event e is enabled at a case C iff $^{\bullet}e \subseteq C$ (input concession) and $e^{\bullet} \cap C = \emptyset$ (output concession)

In a contact-free EN system the input concession suffices for an event to be enabled.

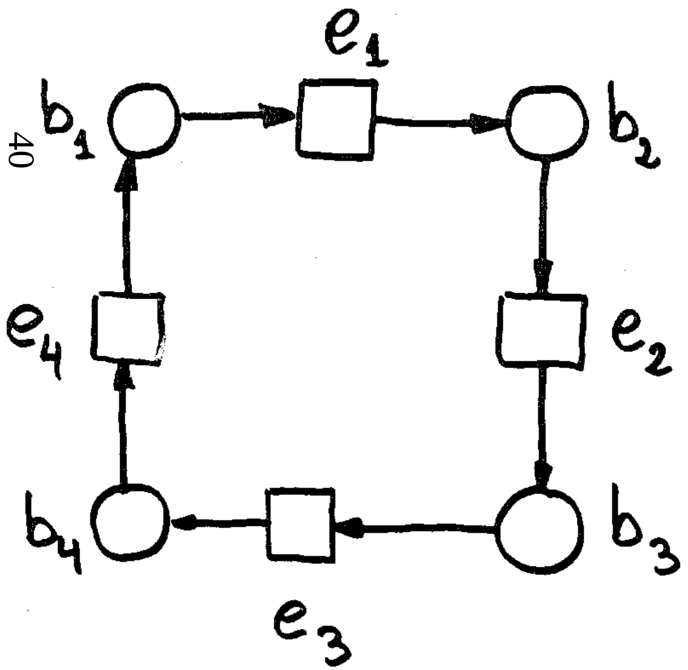
An EN system $\mathcal{N}$ is contact-free iff $(\forall e)_{E_{\mathcal{N}}} (\forall C)_{\mathcal{C}_{\mathcal{N}}}$

$$[\,^{\bullet}e \subseteq C \text{ implies } e^{\bullet} \cap C \neq \emptyset\,]$$

Hence in a contact-free EN system



EXAMPLE

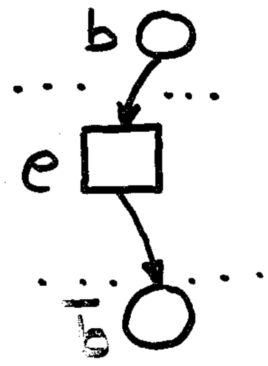


inc($\mathcal{N}$) = $\{b_1, b_3\}$

$\mathcal{N}$ is not contact free

Conditions b_1, b_2 are complements of each other, $b_1 = \overline{b_2}$, iff

$(\forall e \in E)\,[\,b_1 \in {}^\bullet e \text{ iff } b_2 \in e^\bullet\,]$



If $\mathcal{N}$ is (condition) simple then each $b \in B$ has at most one complement.

CONSTRUCTION

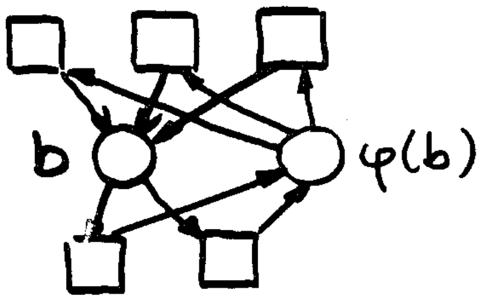
Let $\mathcal{N} = (B, E, F, C_{in})$ be EN system.
Let $\bar{B}$ be a set disjoint with $B \cup E$,
and let $\varphi : B \to \bar{B}$ be a bijection.
The S-complementation of $\mathcal{N}$ (relative to $(\bar{B}, \varphi)$) is the EN system $\mathcal{N}' = (B', E', F', C'_{in})$ such that

$B' = B \cup \bar{B}$, $E' = E$,

$F' = F \cup \{(e, \varphi(b)) : e \in E \ \& \ (b,e) \in F\}$
$\cup \{(\varphi(b), e) : e \in E \ \& \ (e,b) \in F\}$,

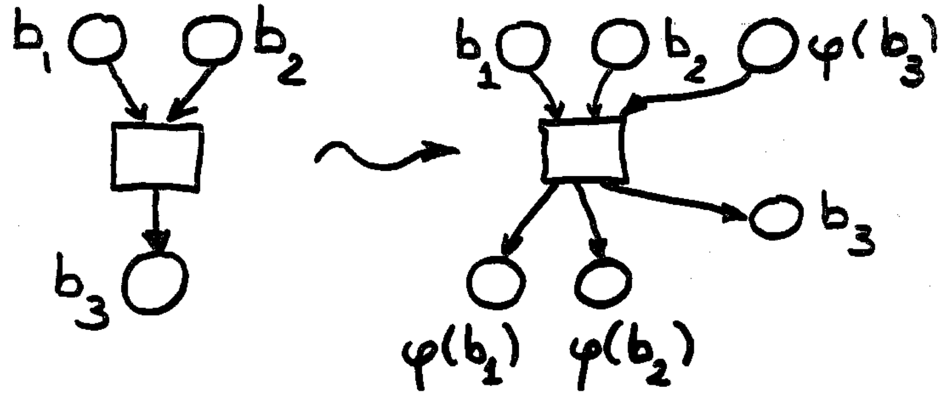
$C'_{in} = C_{in} \cup \varphi(B - C_{in})$.

~·~



If you want to be thrifty you add $\varphi(b)$ only to such $b \in B$ that do not have $\bar{b} \in B$.

It is easier to understand the basic idea of this construction if we assume that no b in B has a complement in B.



BASIC FEATURE:

$$\forall C' \in \mathcal{C}_{\mathcal{N}'} \ \forall b \in B$$
$$[\, b \in C' \text{ iff } \bar{b} \notin C' \,]$$

But

$$\forall e \in E' \ \forall b \in {}^{\bullet}e \ [\, \bar{b} \in e^{\bullet} \,]$$

and so

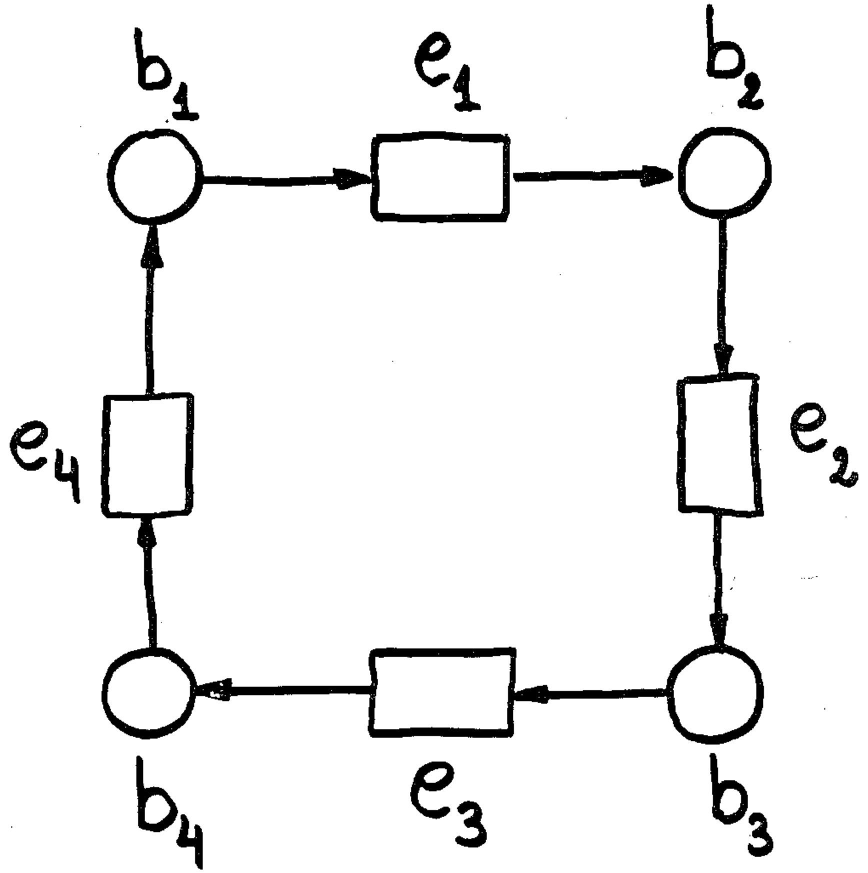
$$\forall C' \in \mathcal{C}_{\mathcal{N}'} \ \forall e \in E'$$
$$[\text{if } {}^{\bullet}e \subseteq C' \text{ then } e^{\bullet} \cap C' = \emptyset]$$

Hence

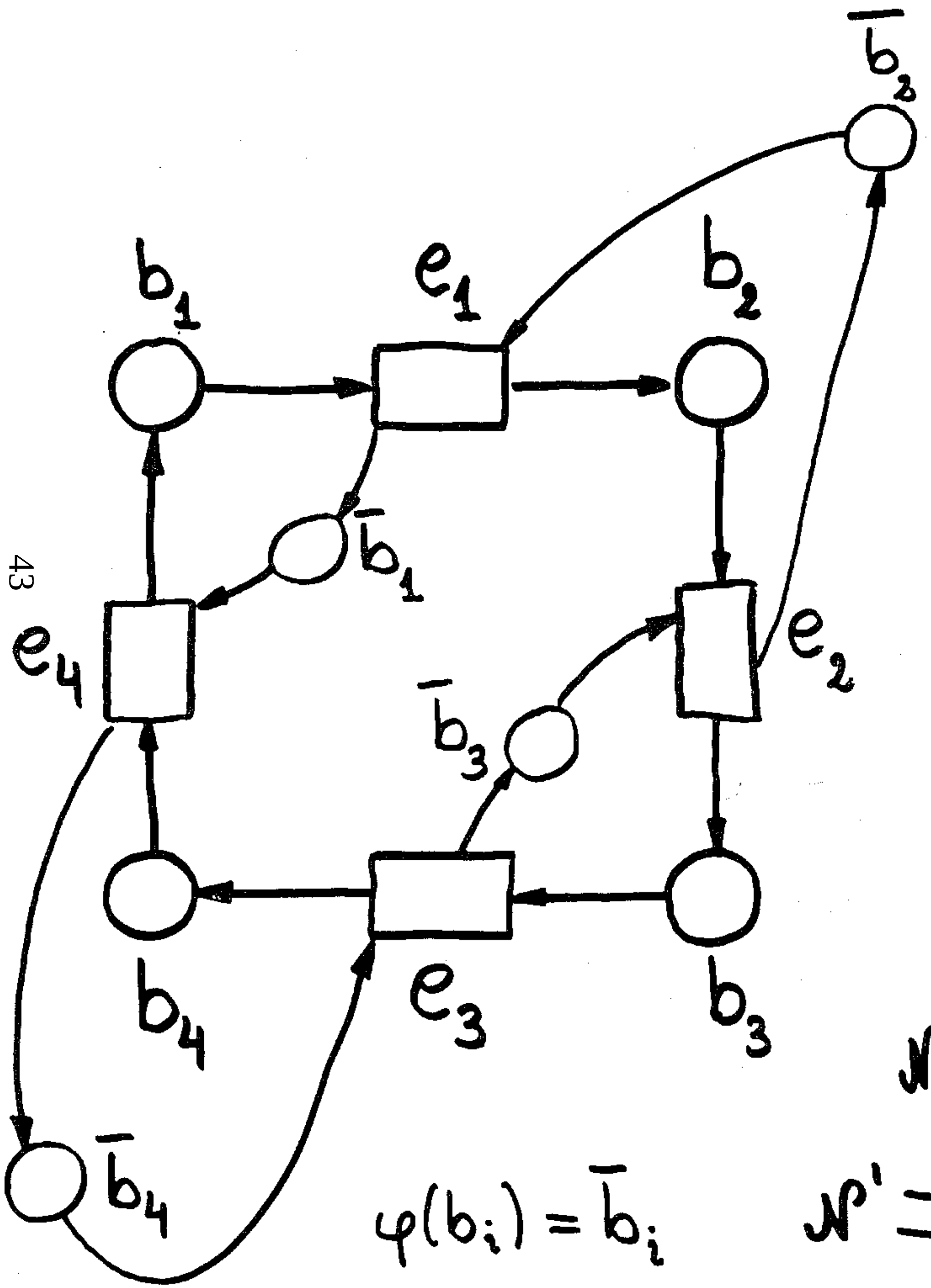
input concession implies output concession

Thus

$\mathcal{N}'$ is contact-free.



$\mathcal{N}$ -



$\varphi(b_i) = \bar{b}_i$

$\mathcal{N}' = \mathcal{N}-$

$\mathcal{N}_1, \mathcal{N}_2$ EN systems.

$\mathcal{N}_2$ is a complementation of $\mathcal{N}_1$ iff $\mathcal{N}_2$ is the S-complementation of $\mathcal{N}_1$ relative to $(\bar{B}, \varphi)$ for some B, φ.

THEOREM

Let $\mathcal{N}_1, \mathcal{N}_2$ be EN systems such that $\mathcal{N}_2$ is a complementation of $\mathcal{N}_1$.

(1) $\mathcal{N}_2$ is contact-free.

(2) $\mathcal{N}_1 \cong \mathcal{N}_2$

THEOREM

For each EN system there exists a state space similar EN system that is contact free.

1)

BEHAVIOUR OF ELEMENTARY NET SYSTEMS

2)

EN SYSTEM

$$\mathcal{N} = (\underbrace{B, E, F}, \underbrace{C_{in}})$$

- B, E, F → underlying static structure (net) — potential dynamics
- C_{in} → initial state (dynamic state space) — actual dynamics

$\mathcal{E}_{\mathcal{N}}$

$\mathcal{U}_{\mathcal{N}}$

3)

ALL IS FINITE !

EN SYSTEMS ARE CONTACT-FREE

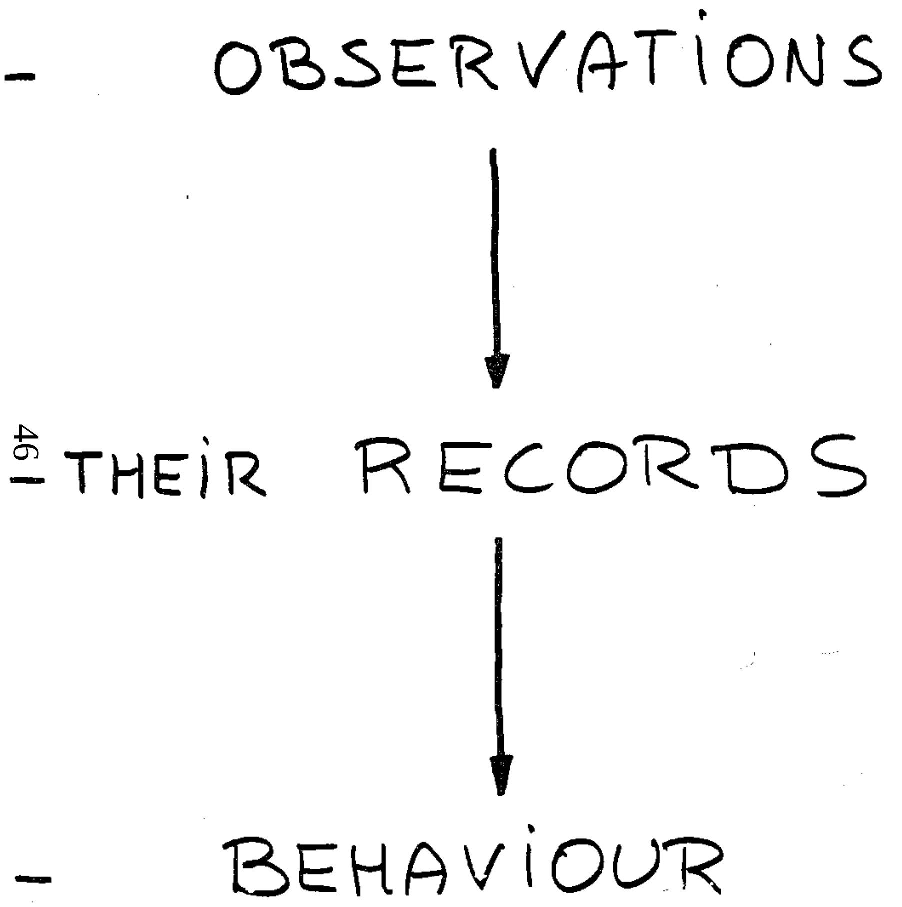
4)

WHAT IS THE BEHAVIOUR ?

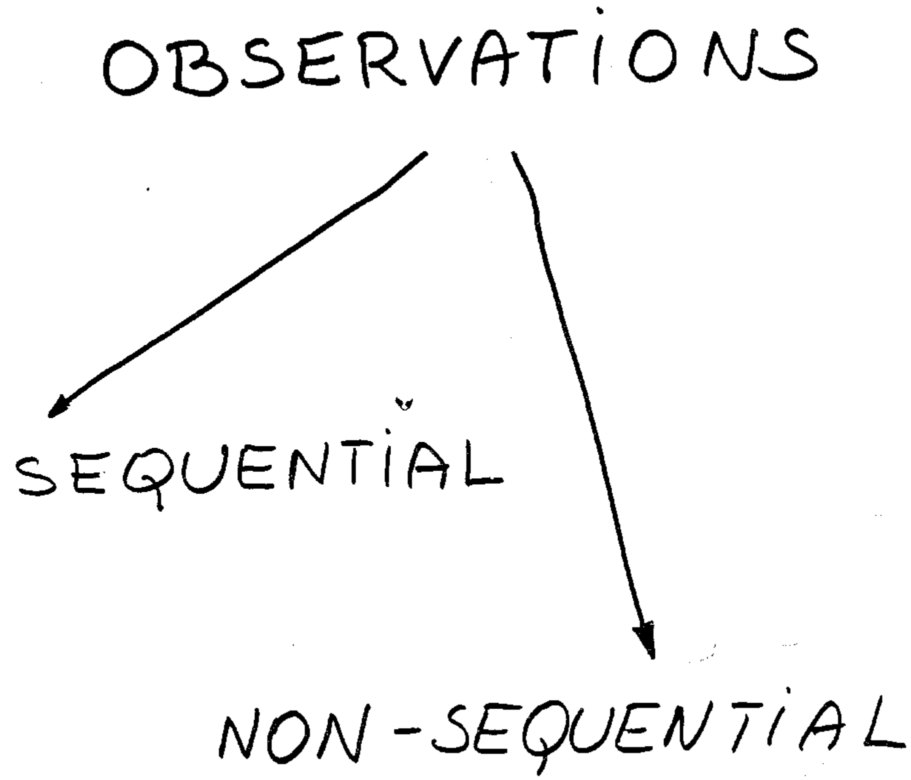
⇕

HOW CAN THE BEHAVIOUR BE OBSERVED ?

5)

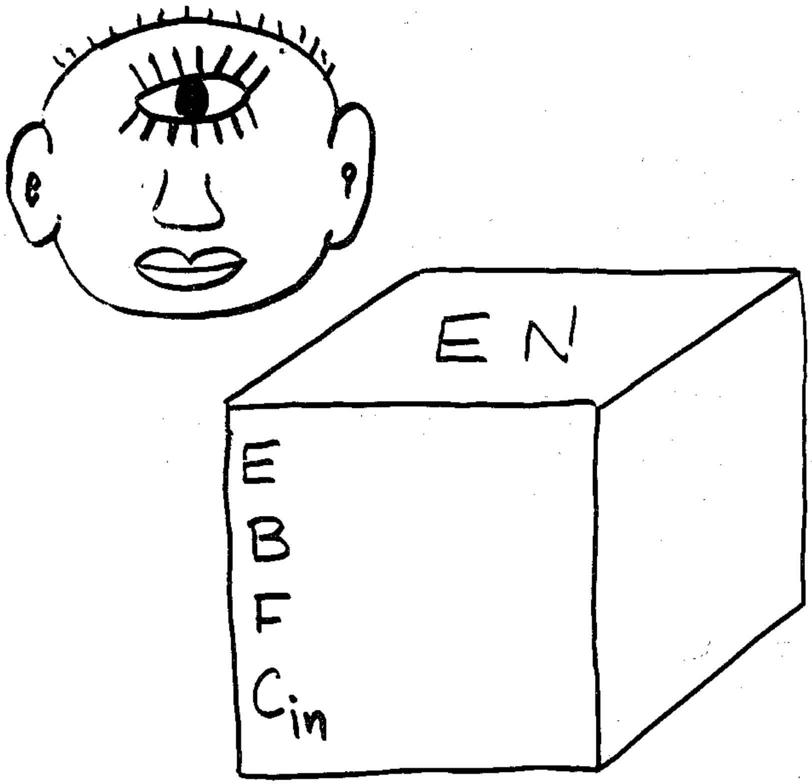

- OBSERVATIONS
- THEIR RECORDS
- BEHAVIOUR

6)


OBSERVATIONS
SEQUENTIAL
NON-SEQUENTIAL

7)

SEQUENTIAL OBSERVATIONS



OBSERVED ARE:

OCCURRENCES OF
<u>SINGLE</u> EVENTS

8)

$\varrho \in E^*$ IS A
FIRING SEQUENCE

IFF

$\varrho = \Lambda$

OR

$\varrho = e_1 \dots e_n \quad n \geq 1$

$e_1, \dots, e_n \in E$

WHERE

$(\exists\, C_0, C_1, \dots, C_n \in \mathcal{C}_{\mathcal{N}})$

$C_0 [e_1\rangle C_1 [e_2\rangle \dots [e_n\rangle C_n$

$\|$

C_{in}

9)

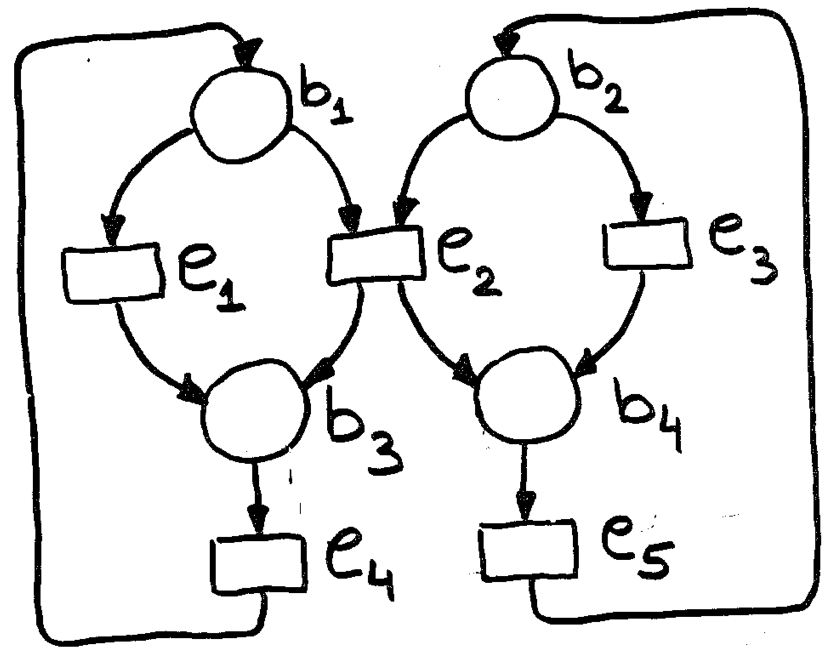
FS$(\mathcal{N}) = FS_{\mathcal{N}}(C_{in})$

TO ANALYZE **FS**$(\mathcal{N})$
WE NEED THE
SEQUENTIAL CASE
GRAPH OF $\mathcal{N}$
SCG$(\mathcal{N})$

OBTAINED BY
DELETING FROM CG$(\mathcal{N})$
ALL EDGES LABELED BY
NON-SINGLETON STEPS

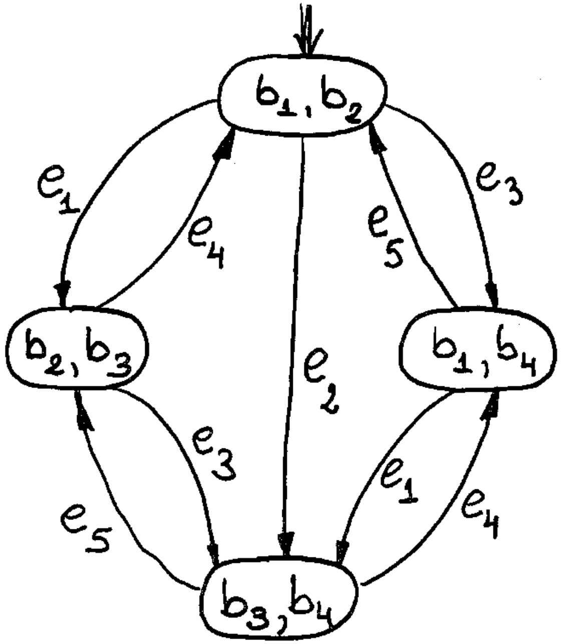
10)

$\mathcal{N}_1$:



$C_{in} = \{b_1, b_2\}$

11)



$SCG(\mathcal{N}_1)$

12)

FS $(\mathcal{N}_1)$:

$e_1\ e_4\ e_2\ e_5\ e_3 \quad \in$

$e_1\ e_3\ e_5\ e_4\ e_1 \quad \in$

$e_1\ e_2\ e_4 \quad \notin$

13)

(1) $\mathbf{FS}(\mathcal{N})$ is PREFIX CLOSED

$$\left.\begin{array}{l} \varrho \in \mathbf{FS}(\mathcal{N}) \\ \varrho = \varrho_1 \varrho_2 \end{array}\right\} \varrho_1 \in \mathbf{FS}(\mathcal{N})$$

(2) $SCG(\mathcal{N})$ is FINITE

<u>THEOREM</u>

$(\forall \mathcal{N})_{EN}$ [$\mathbf{FS}(\mathcal{N})$ is A PREFIX CLOSED REGULAR LANGUAGE] ■

14)

PROBLEMS !!!

$$\ldots\ldots e_1 \, e_3 \ldots\ldots \in$$

IS IT A CAUSAL ORDER?

IS IT ONLY AN OBSERVATIONAL ORDER?

IS $\{e_1, e_3\}$ A STEP?

15)

HOW TO EXTRACT (RECOVER) <u>CAUSAL</u> <u>ORDERS</u> FROM SEQUENTIAL OBSERVATIONS ?

THEORY OF TRACES

(MAZURKIEWICZ)

16)

$\mathcal{N}$ EN SYSTEM

THE INDEPENDENCE RELATION INDUCED BY $\mathcal{N}$: $I_{\mathcal{N}}$

$$(\forall e_1, e_2 \in E_{\mathcal{N}})$$

$$[(e_1, e_2) \in I_{\mathcal{N}} \quad \text{IFF} \quad ({}^\bullet e_1 \cup e_1^\bullet) \cap ({}^\bullet e_2 \cup e_2^\bullet) = \emptyset \;]$$

. . . . .

THE DEPENDENCE RELATION INDUCED BY $\mathcal{N}$: $D_{\mathcal{N}}$

$$D_{\mathcal{N}} = (E_{\mathcal{N}} \times E_{\mathcal{N}}) - I_{\mathcal{N}}$$

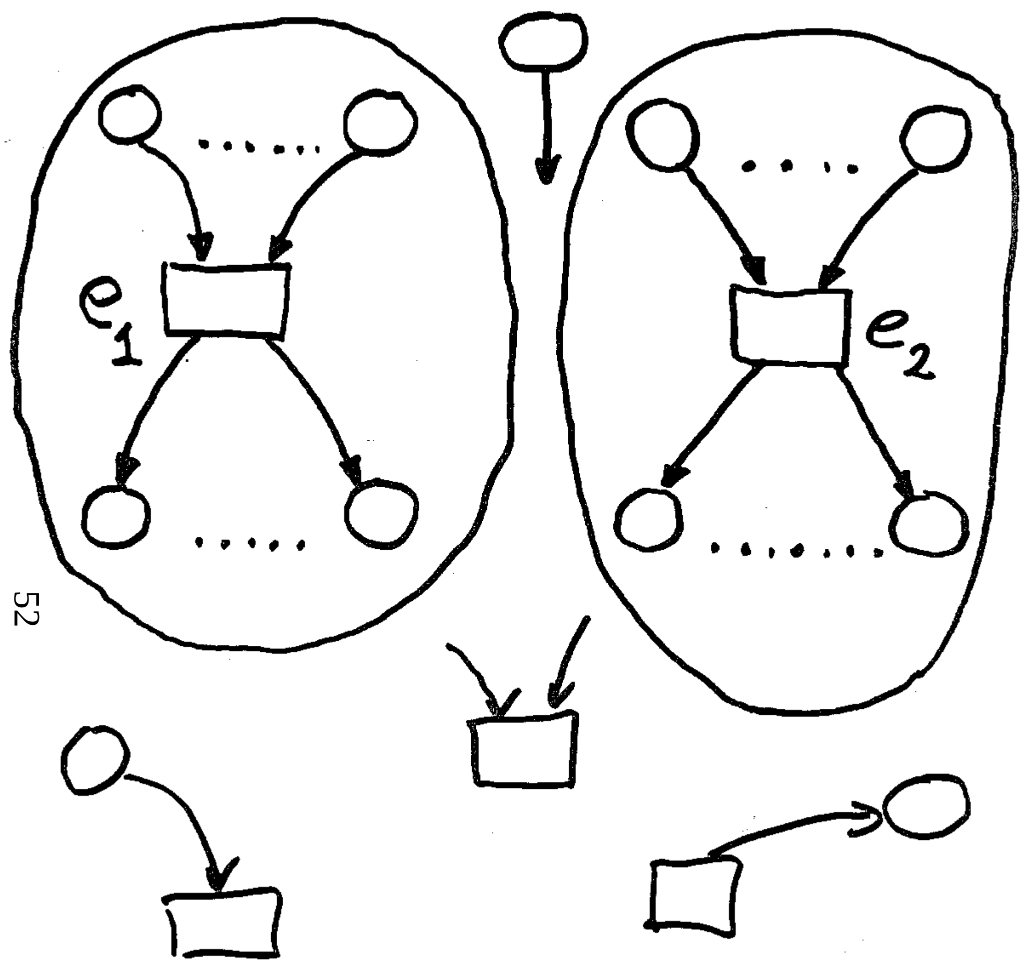
. . . .

$I_{\mathcal{N}}$ IS SYMMETRIC & IRREFLEXIVE

$D_{\mathcal{N}}$ IS SYMMETRIC & REFLEXIVE

②

17)



$$(e_1, e_2) \in I_{\mathcal{N}}$$

③

18)

$$\varrho = \ldots\ldots e_1 e_2 \ldots\ldots \in E^*_{\mathcal{N}}$$

$$\mu = \ldots\ldots e_2 e_1 \ldots\ldots$$

$$\frac{(e_1, e_2) \in I_{\mathcal{N}}}{\varrho \doteq_{I_{\mathcal{N}}} \mu}$$

- - - - - - - - - - - - - - - -

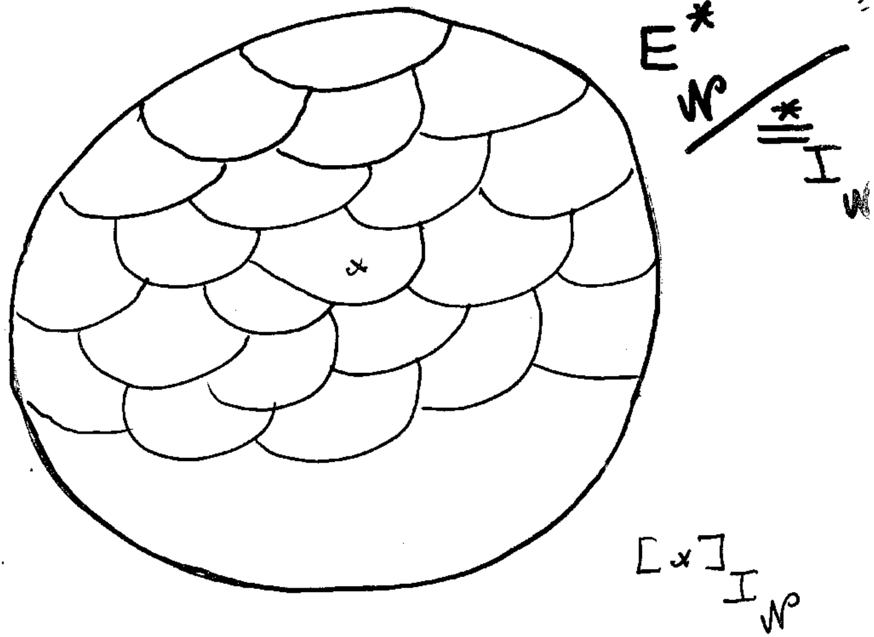
$$\frac{\varrho \doteq_{I_{\mathcal{N}}} \mu \doteq_{I_{\mathcal{N}}} \gamma \ldots \doteq_{I_{\mathcal{N}}} \delta}{\varrho \overset{*}{=}_{I_{\mathcal{N}}} \delta}$$

- - - - - - - - - - - - - - - -

$\overset{*}{=}_{I_{\mathcal{N}}}$ an equivalence relation on $E^*_{\mathcal{N}}$

4

14



$Z \subseteq E^*_{\mathcal{N}}$ is **$I_{\mathcal{N}}$-CONSISTENT**

IFF Z IS A UNION OF

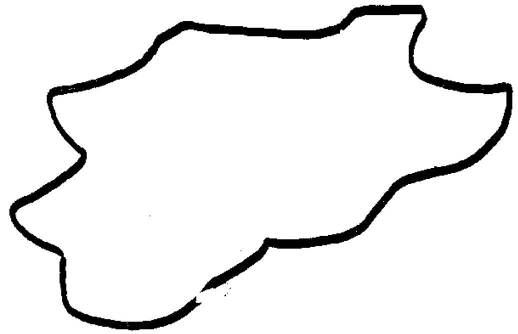
EQUIVALENCE CLASSES

OF $\overset{*}{\equiv}_{I_{\mathcal{N}}}$.

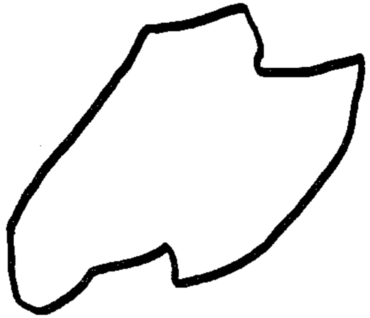
An equivalence class is called a **trace**

4'

20)



This Z is $I_{\mathcal{N}}$-CONSISTENT



This Z is **NOT** $I_{\mathcal{N}}$-CONSISTENT

(5)

THEOREM

$(\forall \mathcal{N})_{EN}$ [**FS**$(\mathcal{N})$ is $I_{\mathcal{N}}$-CONSISTENT.]

IF ϱ OBSERVABLE IN $\mathcal{N}$,
THEN EACH ELEMENT OF
$[\varrho]_{I_{\mathcal{N}}}$ OBSERVABLE IN $\mathcal{N}$.

Each equivalence class $[\alpha]_{I_{\mathcal{N}}}$ *is either* ***included*** *in* **FS**$(\mathcal{N})$ *or* ***disjoint*** *with* **FS**$(\mathcal{N})$

Those that are included are called
(FIRING) TRACES

FT$(\mathcal{N})$

SEQUENTIAL OBSERVATION

- FIRING SEQUENCES

Linear – difficult to interpret

break them down to

- DEPENDENCE GRAPHS

acyclic directed graphs

⇝

- PARTIAL ORDERS

(7) $\mathcal{N}$ $\qquad \varrho \in \mathbf{FS}(\mathcal{N})$

THE CANONICAL DEPENDENCE GRAPH OF ϱ $\quad \langle \varrho \rangle_{D_{\mathcal{N}}}$

(i) $\varrho = \lambda \rightsquigarrow \langle \varrho \rangle_{D_{\mathcal{N}}}$ is empty

(ii) $\varrho = e_1 \dots e_n$, $n \geq 1$, $e_1, \dots, e_n \in E_{\mathcal{N}}$ $\rightsquigarrow$

$\langle \varrho \rangle_{D_{\mathcal{N}}}$ is the $E_{\mathcal{N}}$-lab. graph (V, Y, φ)

- $V = \{1, \dots, n\}$,
- $(\forall i \in \{1, \dots, n\})\ [\varphi(i) = e_i]$
- $(\forall i, j \in \{1, \dots, n\})$

  $[(i, j) \in Y$ iff

  $(i < j)$ & $(e_i, e_j) \in D_{\mathcal{N}}]$

25)

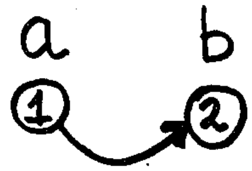
$$\varrho = a\ b\ c\ a\ d$$

a

①

26)

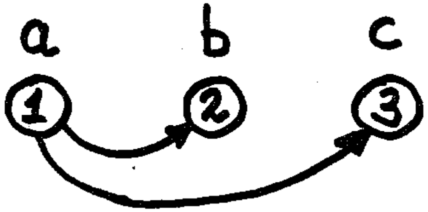
$$\varrho = a\ b\ c\ a\ d$$



$$(a, b) \in D$$

27)

$\varrho = a\ b\ c\ a\ d$

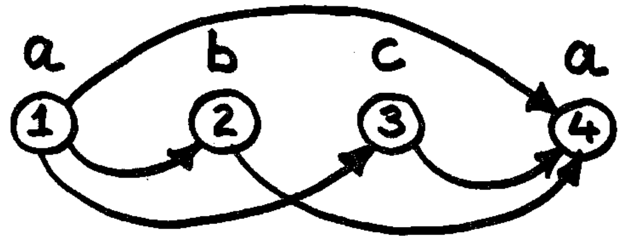


$(c, a) \in D$ $(c, b) \in I$

$(a, b) \in D$

28)

$\varrho = a\ b\ c\ a\ d$

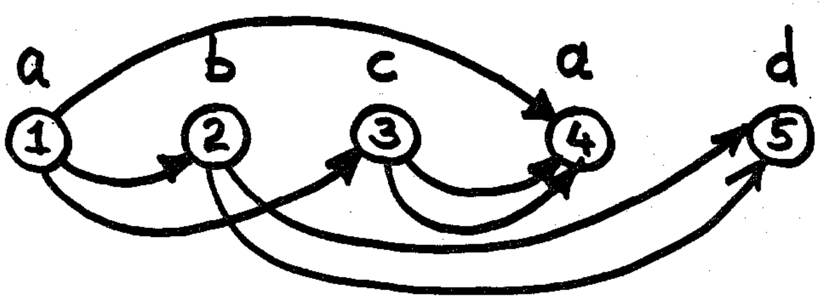


$(a, a) \in D$

$(c, a) \in D$ $(c, b) \in I$

$(a, b) \in D$

$\varrho = a\ b\ c\ a\ d$

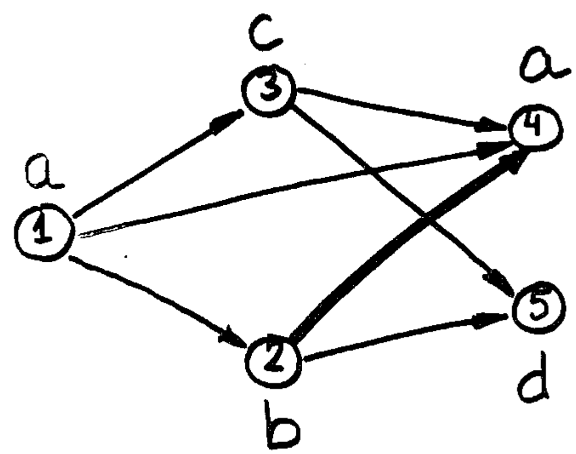


$(d,b) \in D$ $(d,c) \in D$ $(d,a) \in I$

$(a,a) \in D$

$(c,a) \in D$ $(c,b) \in I$

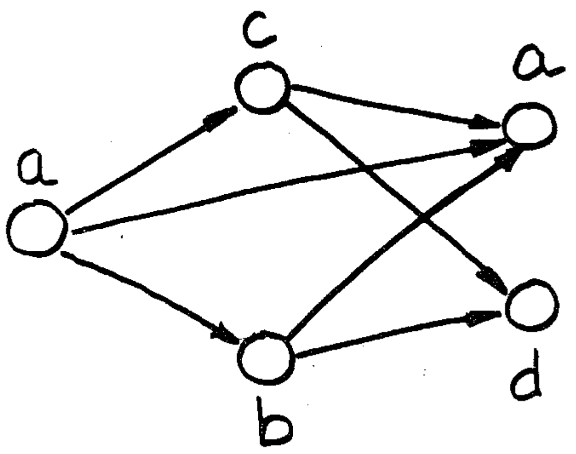
$(a,b) \in D$



$\langle \varrho \rangle_D$ the canonical dependence graph of ϱ

$\varrho = a\ b\ c\ a\ d$

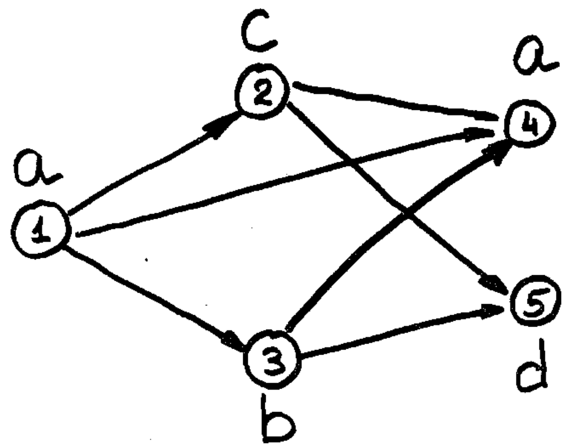
31)



$\overline{\langle \varrho \rangle}_D$

<u>abstract</u>
dependence graph
of ϱ

32)



$\langle \varrho' \rangle_D$

$\varrho' = a\,c\,b\,a\,d$

$\varrho = a\,\underline{b\,c}\,a\,d \qquad (b,c) \in I$

$\varrho' = a\,\underline{c\,b}\,a\,d$

$\langle \varrho' \rangle_D$ isom $\langle \varrho \rangle_D$

So

$\overline{\langle \varrho' \rangle_D} = \overline{\langle \varrho \rangle_D}$

THEOREM

$\mathcal{N} = (B, E, F, C_{in})$ EN system,

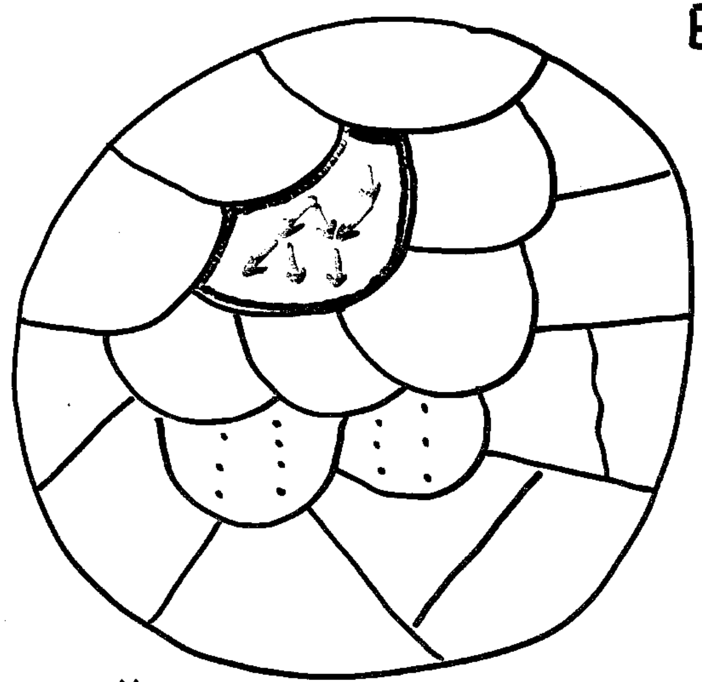
$\varrho_1, \varrho_2 \in E^*$.

$\varrho_1 \overset{*}{\equiv} \varrho_2$ iff $\langle \varrho_1 \rangle_D$ isom $\langle \varrho_2 \rangle_D$

iff $\overline{\langle \varrho_1 \rangle_D} = \overline{\langle \varrho_2 \rangle_D}$

iff $[\varrho_1]_I = [\varrho_2]_I$

$E^* / \stackrel{*}{\equiv}_I$



$t \in E^*$

$t =$  $\longrightarrow$  $= \overline{\langle t \rangle}_D$

every string from t will yield the same abstract dependence graph

$\overline{\langle t \rangle}_D$ - abstract dependence graph of t. <u>adg</u>(t)

Thus, if t is a firing trace,
i.e., $t = [\varrho]_I$ for some $\varrho \in FS(\mathcal{N})$,
then
each firing sequence from t is a sequential observation of (in general nonsequential) "run" ("process", ...) of $\mathcal{N}$.

Hence
each firing trace is the set of <u>all</u> sequential observations of <u>the same</u> run of $\mathcal{N}$.

35)

POSETS

(A, R) ANTISYM.
TRANSIT.
REFLEX.

- - - - - - - - - - - - - - - - - - - -

$g = (V, E)$ DIR. ACYCL. GRAPH

$\rightsquigarrow$ TRANS. & REFL. CLOS.

$\leqslant_g = (V, E^*)$

36)

$\varrho \in \Sigma^*$

$\langle \varrho \rangle_D$ $\quad \langle\langle \varrho \rangle\rangle_D$

$\overline{\langle \varrho \rangle}_D$ $\quad \overline{\langle\langle \varrho \rangle\rangle}_D$

$\underline{adg}(t)$ $\quad \underline{alp}(t)$

ADG(T) $\quad$ **ALP**(T)

37)

$$\mathbf{FS}(\mathcal{N}) \subseteq E^*$$

$$[\mathbf{FS}(\mathcal{N})]_{I_\mathcal{N}}$$

FIRING TRACES OF $\mathcal{N}$

$\mathbf{FT}(\mathcal{N})$

$\mathbf{ADG}(\mathbf{FT}(\mathcal{N}))$

ABSTRACT FIRING DEP. GRAPHS OF $\mathcal{N}$

$\mathbf{AFD}(\mathcal{N})$

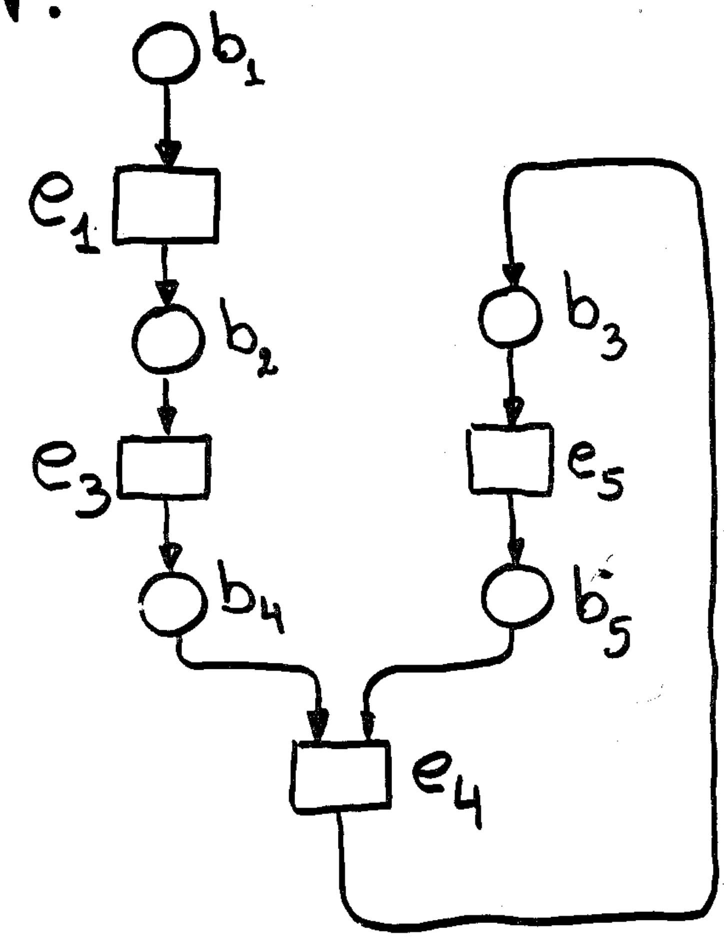
$\mathbf{ALP}(\mathbf{FT}(\mathcal{N}))$

ABSTRACT FIRING LAB. POSETS OF $\mathcal{N}$

$\mathbf{AFLP}(\mathcal{N})$

38)

$\mathcal{N}$:



$$C_{in} = \{b_1, b_3\}$$

39)

$$I_{\mathcal{N}} = \{ (e_1, e_5), (e_5, e_1), (e_1, e_4), (e_4, e_1), (e_3, e_5), (e_5, e_3) \}$$

$$D_{\mathcal{N}} = E \times E - I_{\mathcal{N}}$$

40)

$$\varrho = e_1 e_5 e_3 e_4 e_5 \in \mathbf{FS}(\mathcal{N})$$

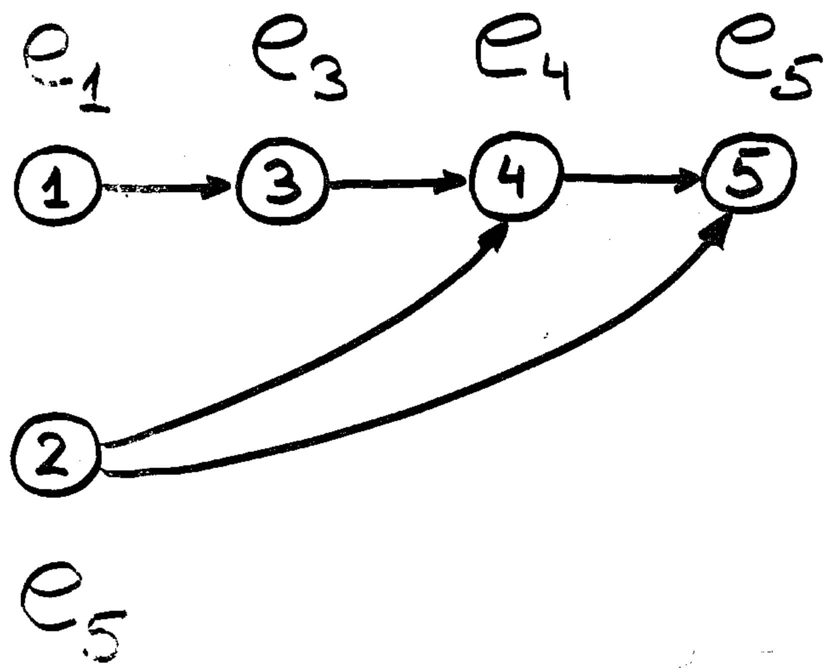
$$[\varrho]_{I_{\mathcal{N}}} = \{ e_1 e_5 e_3 e_4 e_5, \ e_5 e_1 e_3 e_4 e_5, \ e_1 e_3 e_5 e_4 e_5 \}$$

41)

$$[\varrho]_{I_{\mathcal{N}}} \subseteq \mathbf{FS}(\mathcal{N})$$

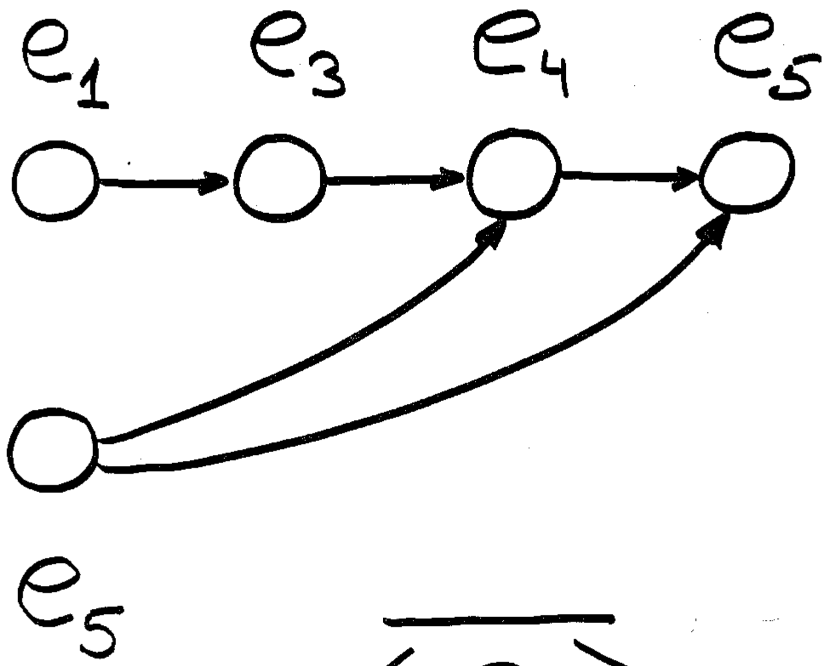
$$[\varrho]_{I_{\mathcal{N}}} \in \mathbf{FT}(\mathcal{N})$$

42)



$$\langle \varrho \rangle_{D_{\mathcal{N}}}$$

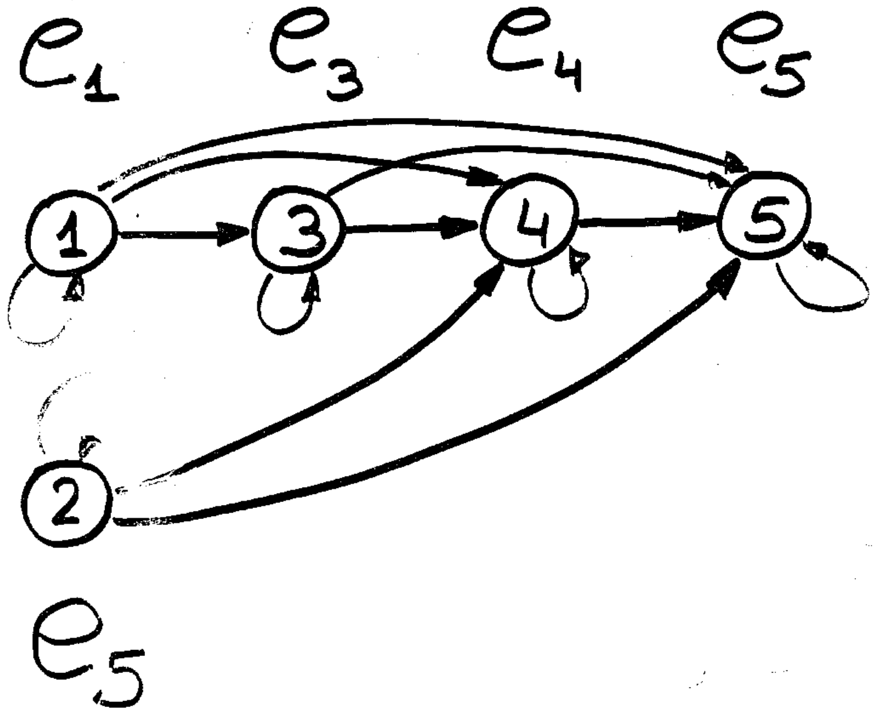
43)



$$\overline{\langle e \rangle}_{D_{\mathcal{N}}}$$

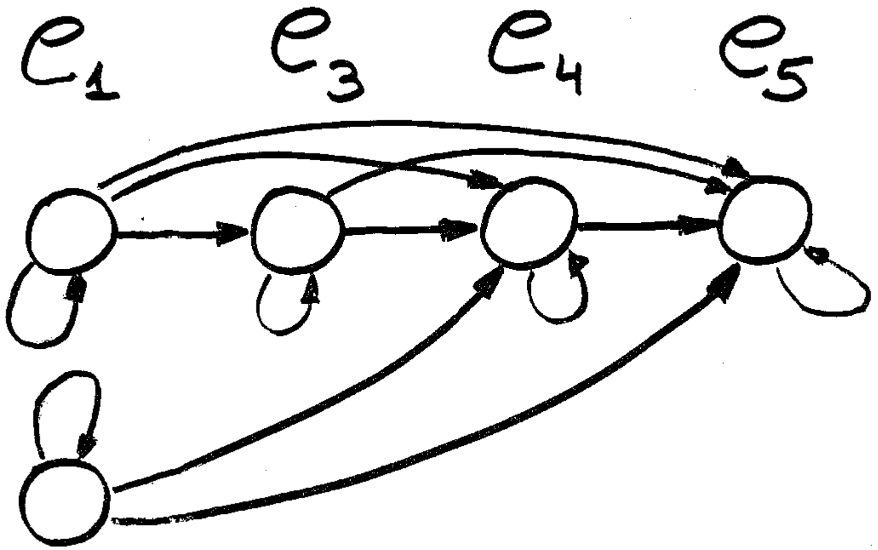
$$\in \mathbf{AFD}(\mathcal{N})$$

44)



$$\langle\langle e \rangle\rangle_{D_{\mathcal{N}}}$$

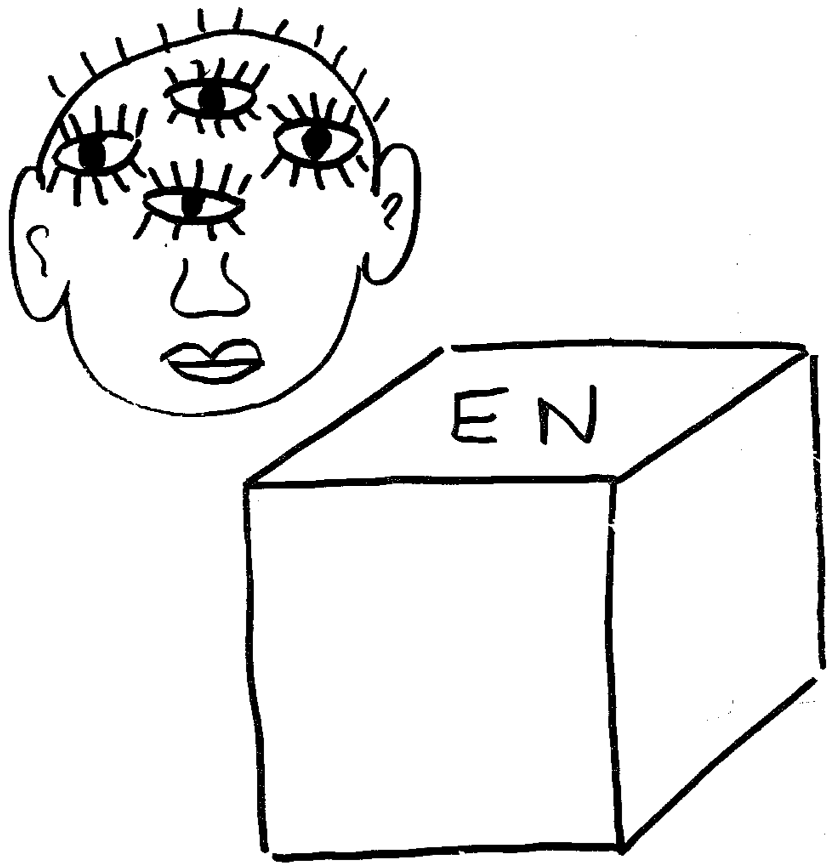
45)



$$\overline{\langle\langle e \rangle\rangle}_{D_{\mathcal{N}}}$$

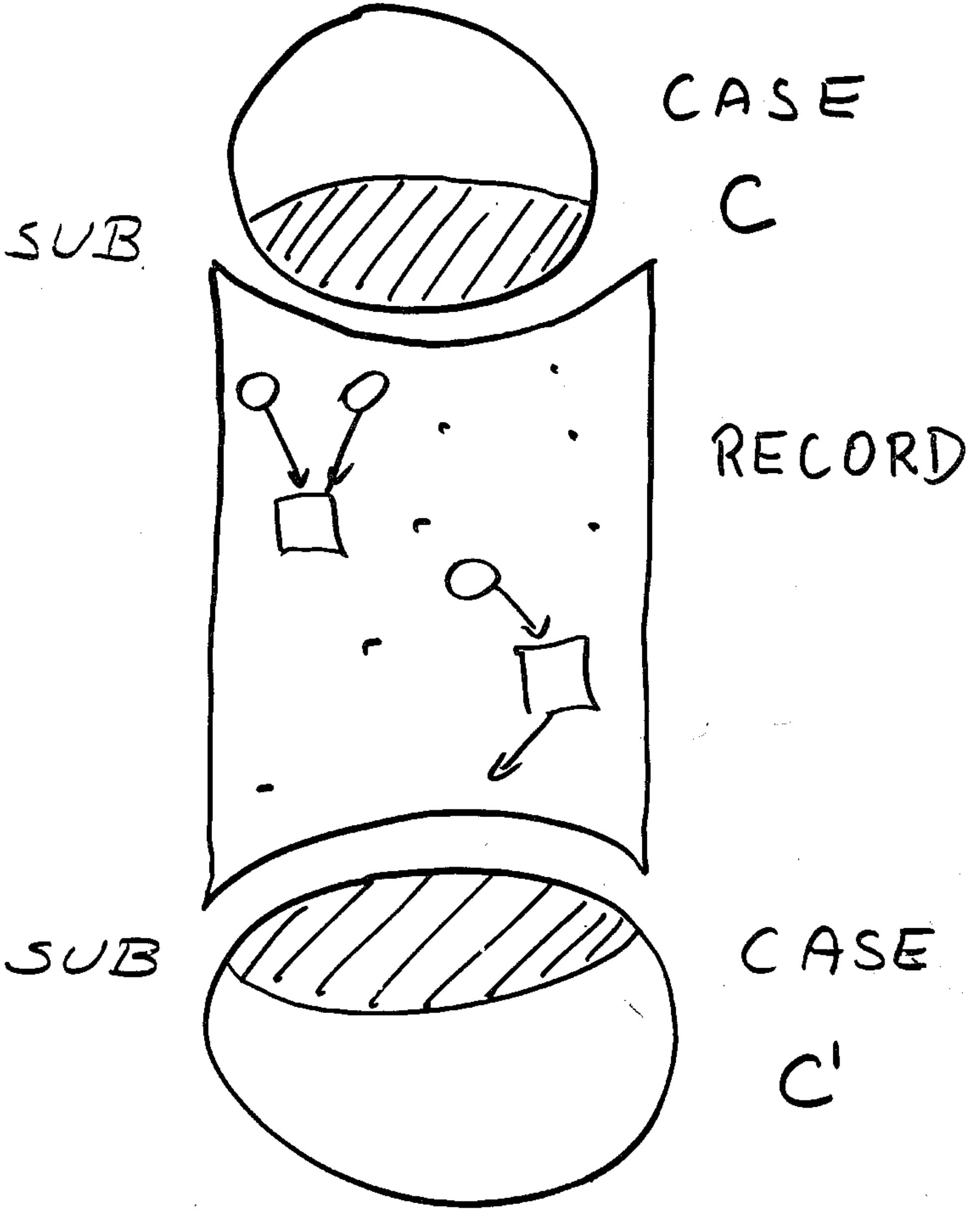
$\in$ **AFLP**$(\mathcal{N})$

46)

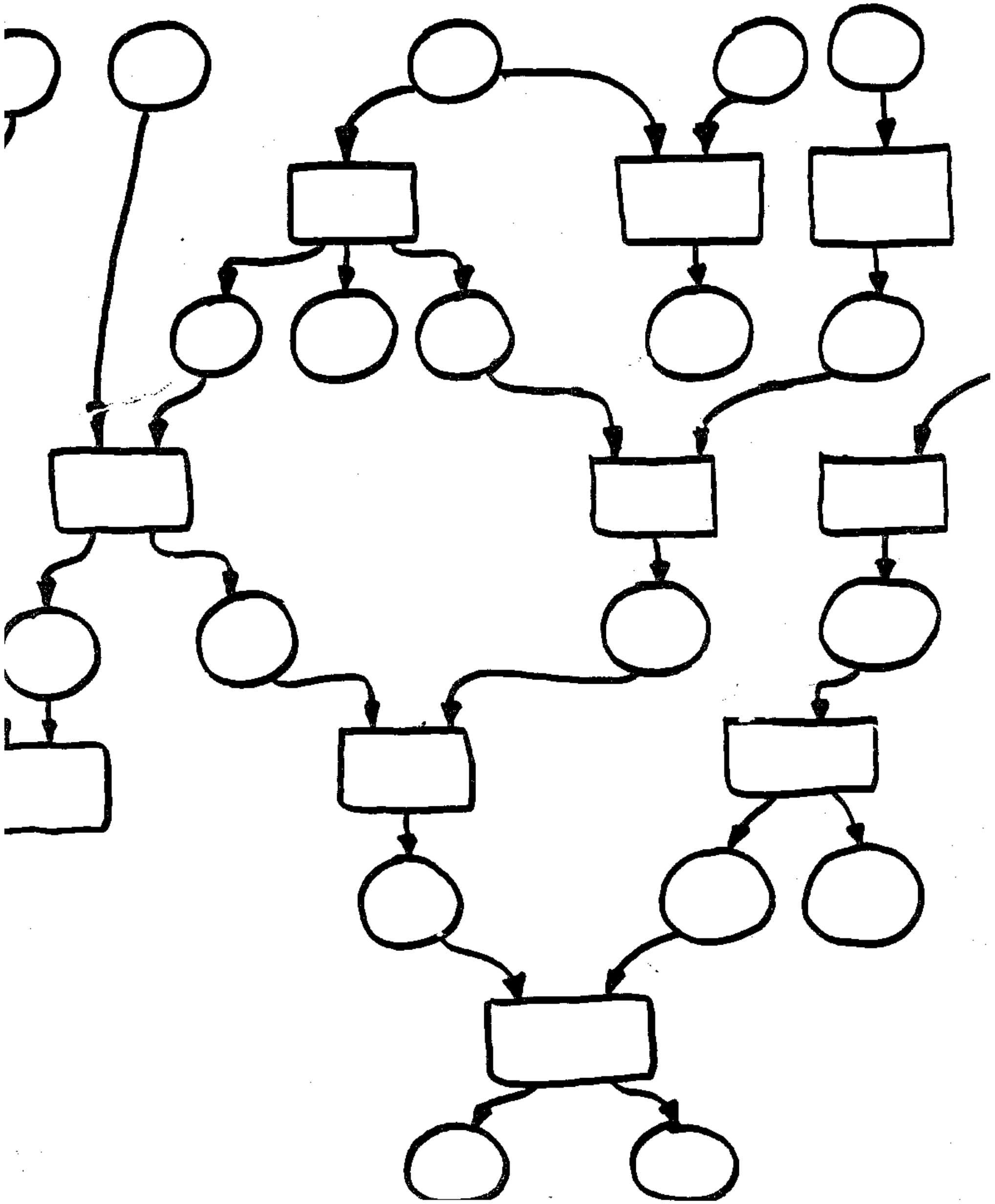


NON-SEQUENTIAL OBSERVATIONS

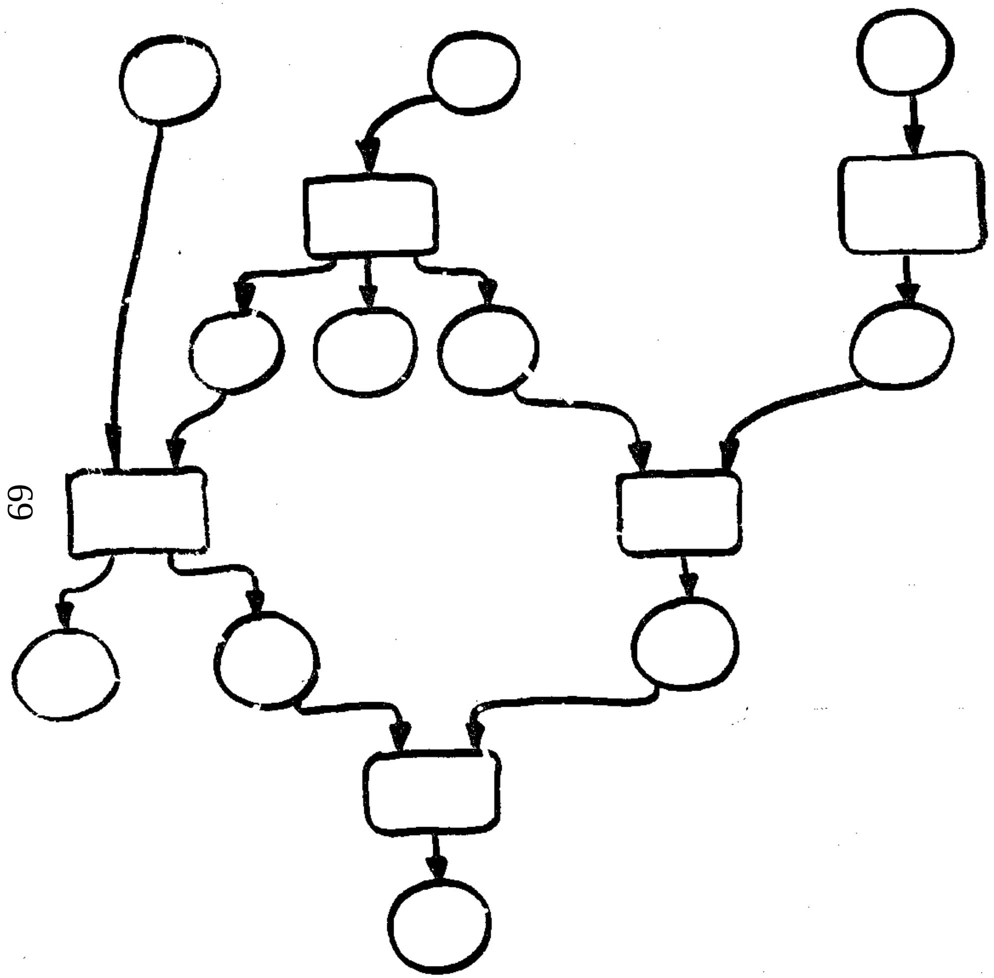
47)



48)

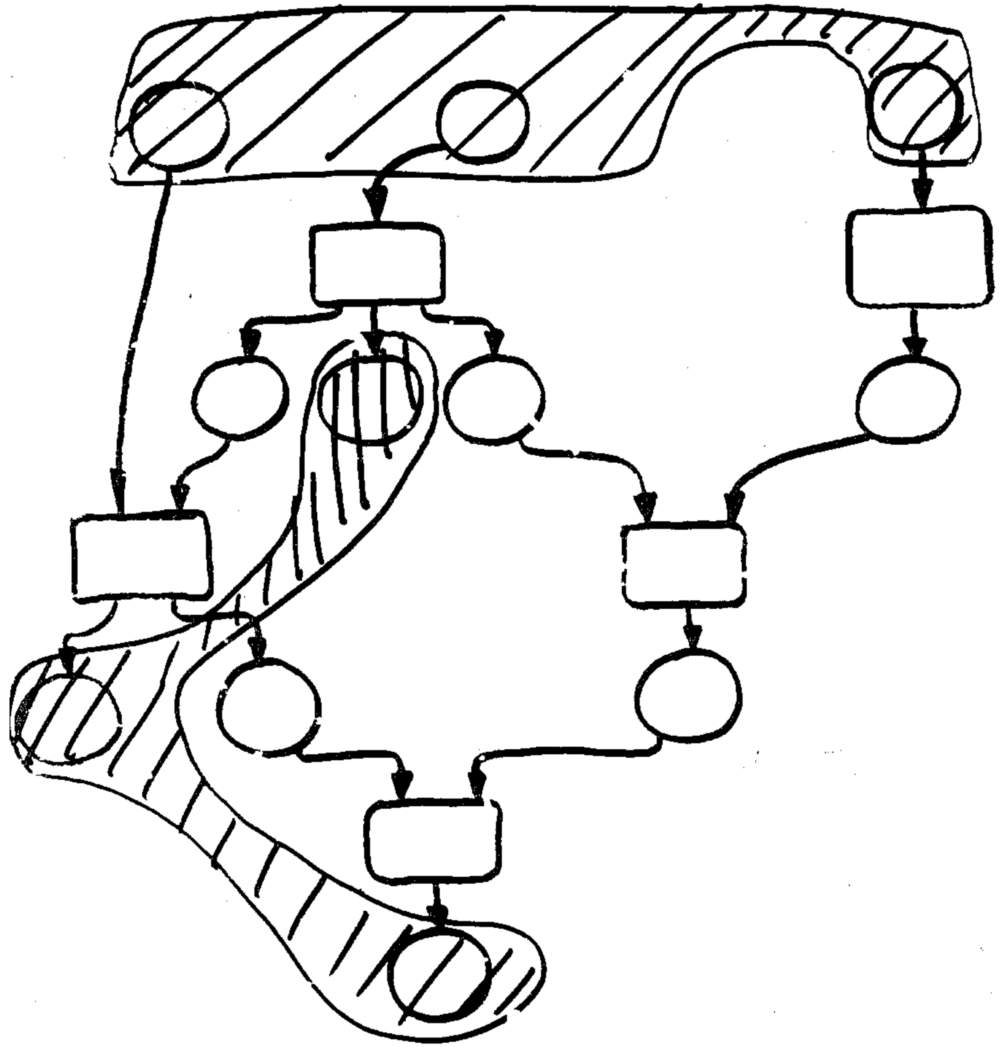


49)

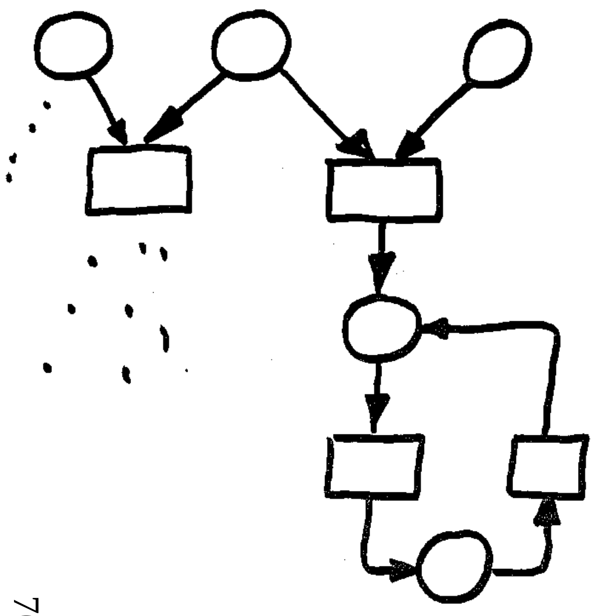


no conflicts in a run!

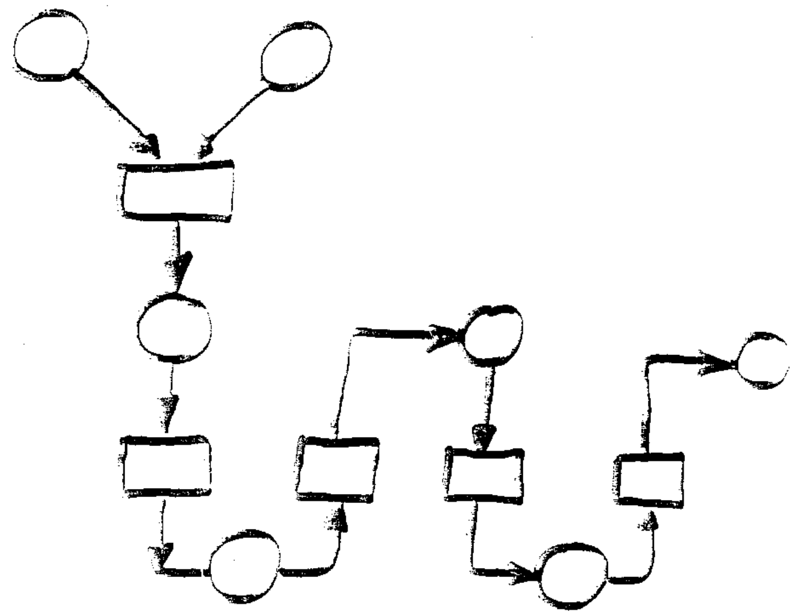
50)



50



50²



no cycles in a run

51)

A NET $N=(S,T,F)$
IS AN OCCURRENCE
NET IFF

$(\forall s \in S)[\,|{}^{\bullet}s| \leq 1$ AND
$|s^{\bullet}| \leq 1\,]$

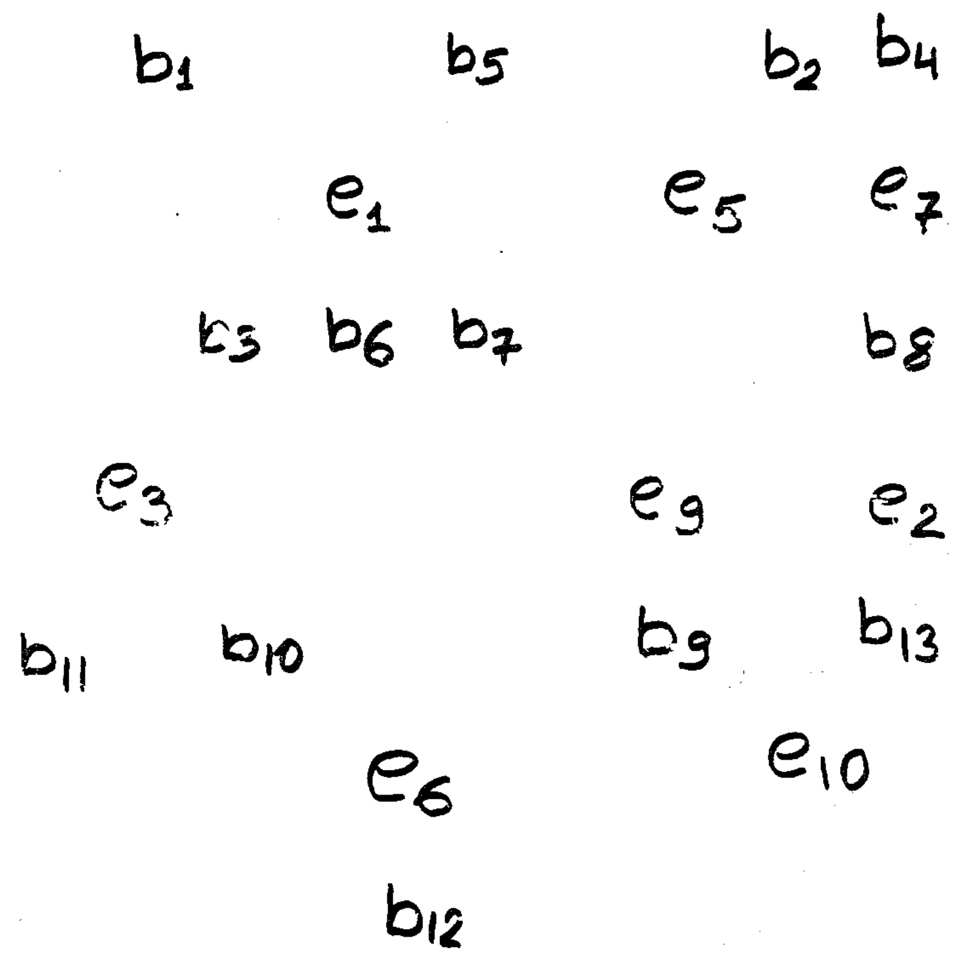
S-NON-BRANCHING

$(\forall x,y \in X)[(x,y) \in F^{+} \Longrightarrow$
$(y,x) \notin F^{+}]$

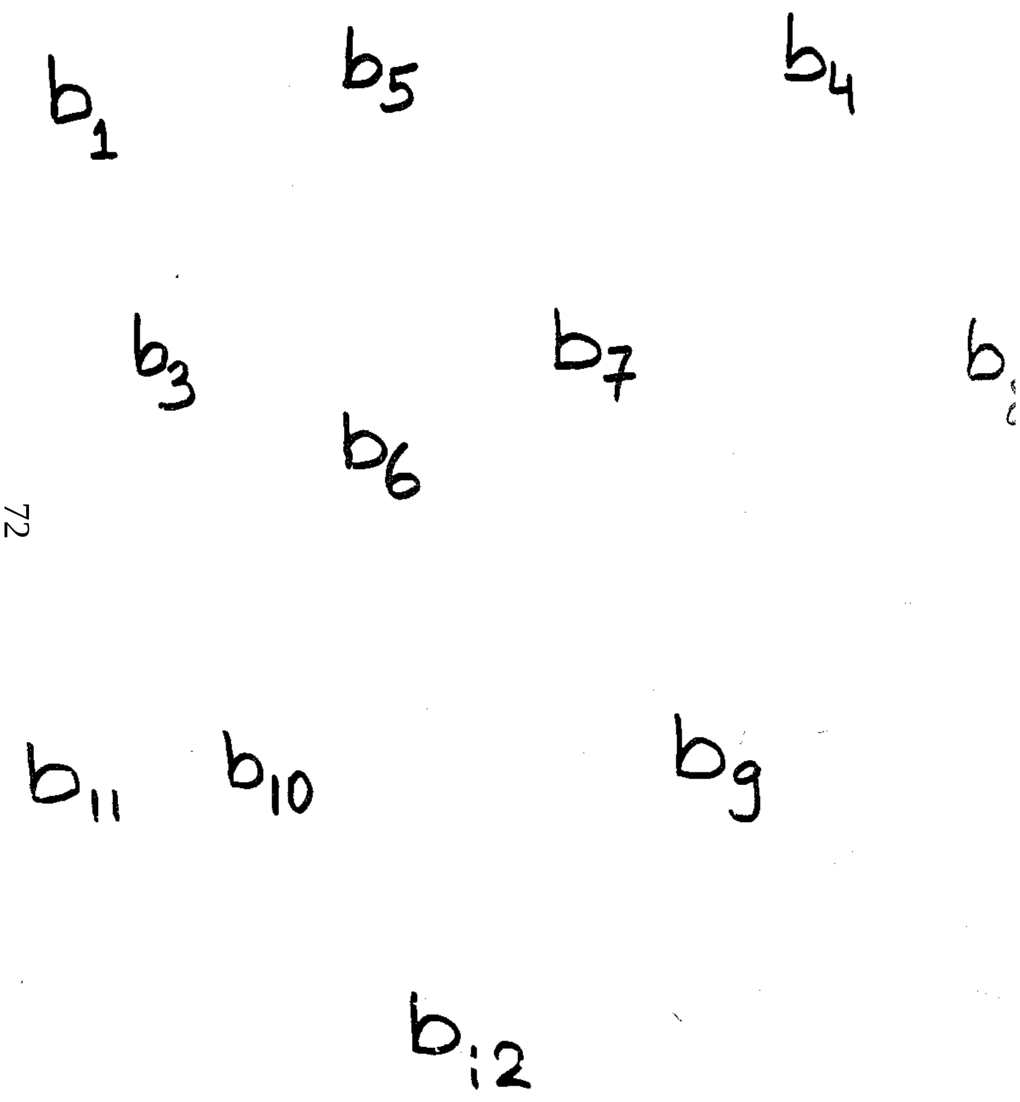
ACYCLIC

$(\forall t \in T)[\,t^{\bullet} \neq \emptyset\,]$

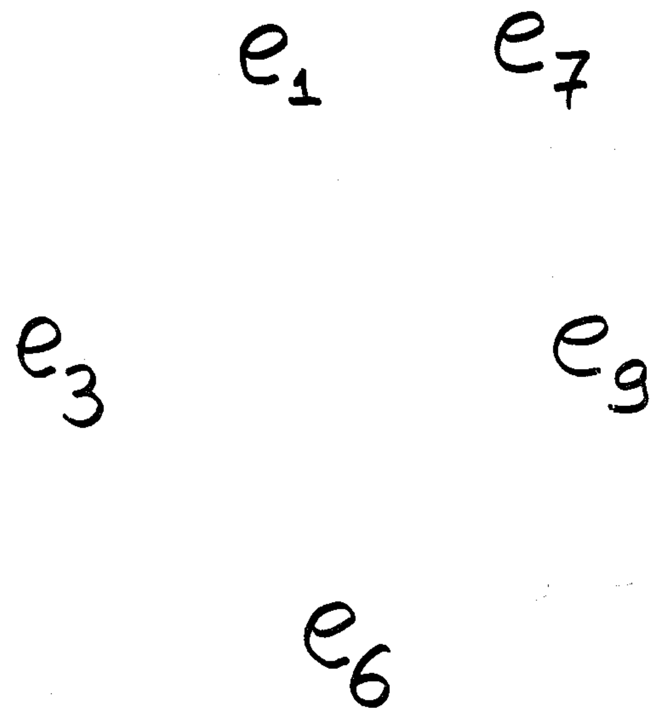
52)



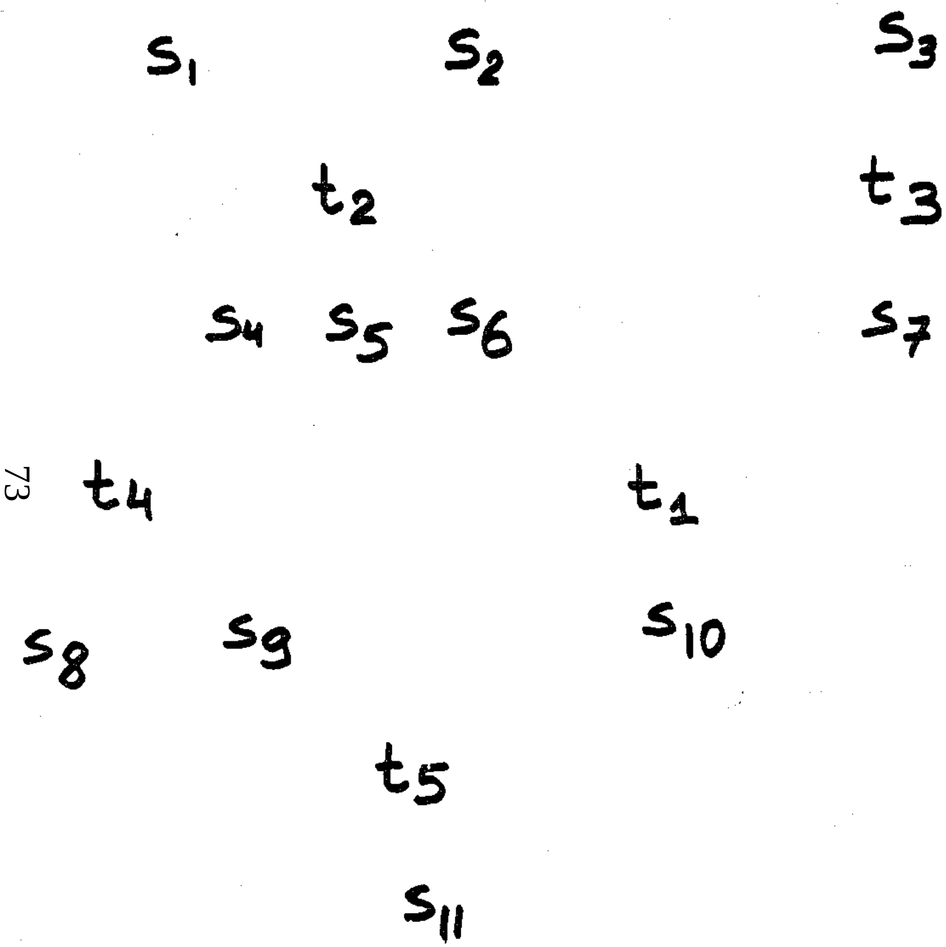
53)



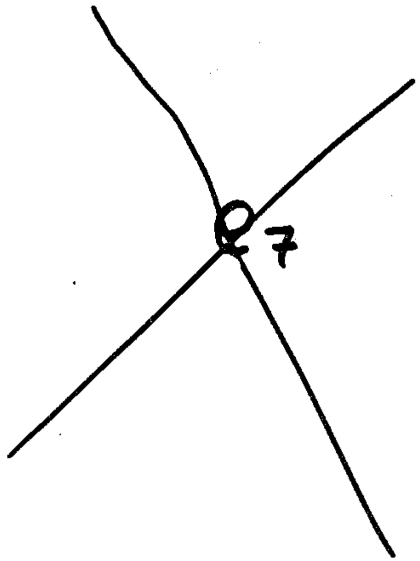
54)



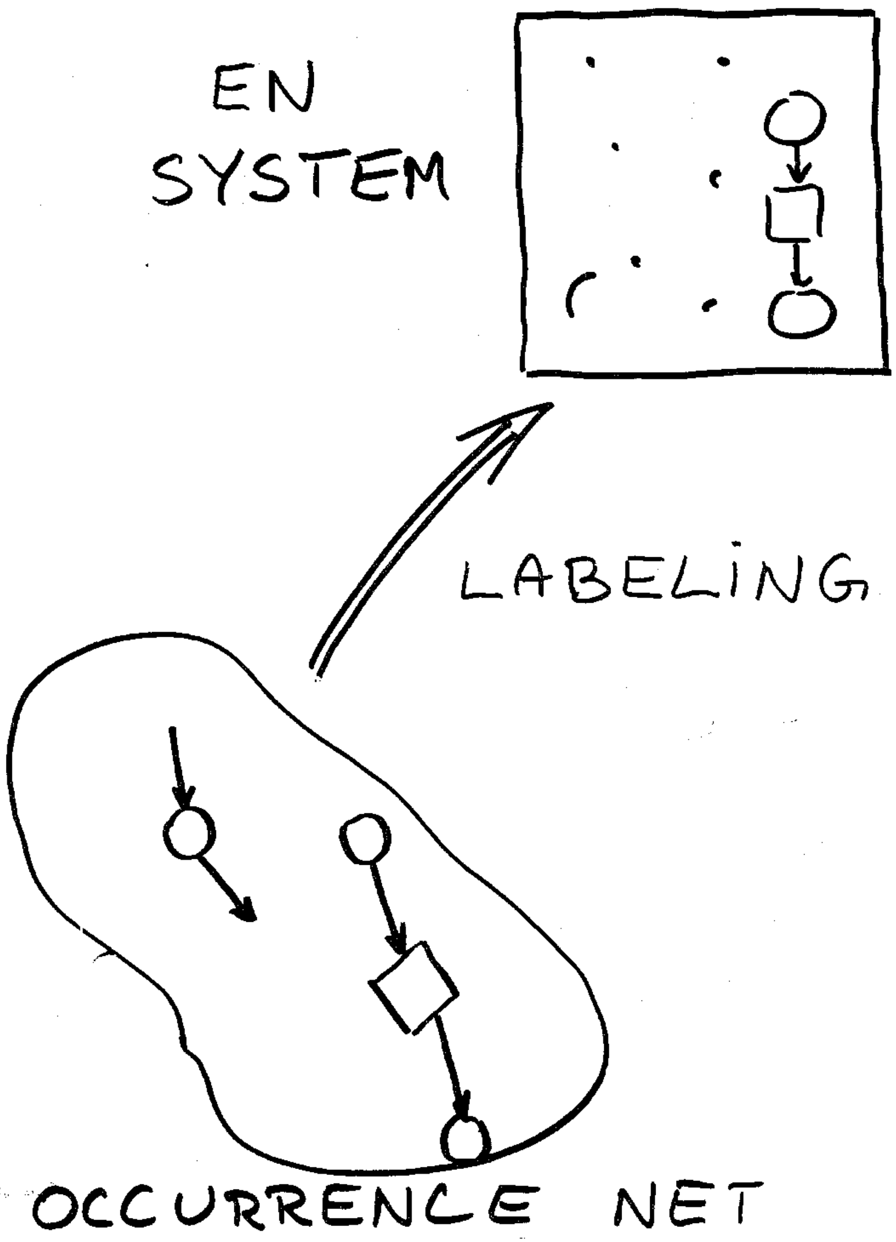
55)



56)



57)



58)

A NODE-LABELED OCCURRENCE NET

$N = (S, T, F, \varphi)$

(S, T, F) OCCURR. NET

$\varphi : S \cup T \to \Sigma$

$\varphi(S) \cap \varphi(T) = \emptyset$

$\mathcal{N}$ EN SYSTEM

$N = (S, T, H, \varphi)$ NODE-LAB. OCCUR. NET

N IS A PROCESS OF $\mathcal{N}$ IFF

(i) $\varphi(S) \subseteq B_{\mathcal{N}}$ AND $\varphi(T) \subseteq E_{\mathcal{N}}$.

(ii) $(\forall s_1, s_2 \in S)\,[\varphi(s_1) = \varphi(s_2)$

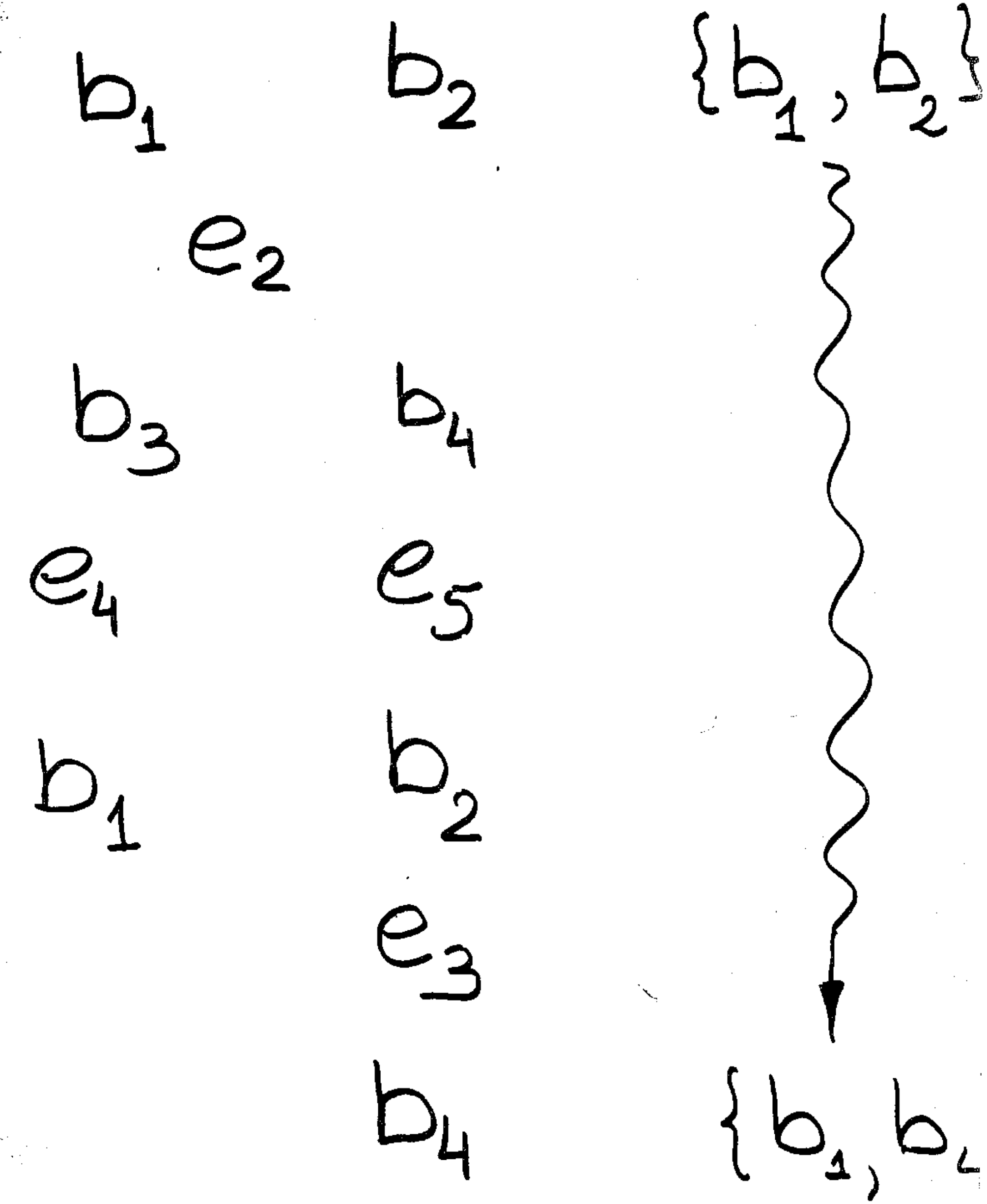
$\Longrightarrow (s_1 \leq_N s_2) \vee (s_2 \leq_N s_1)]$

(iii) $(\forall t \in T)\,[\varphi({}^{\bullet}t) = {}^{\bullet}\varphi(t)$

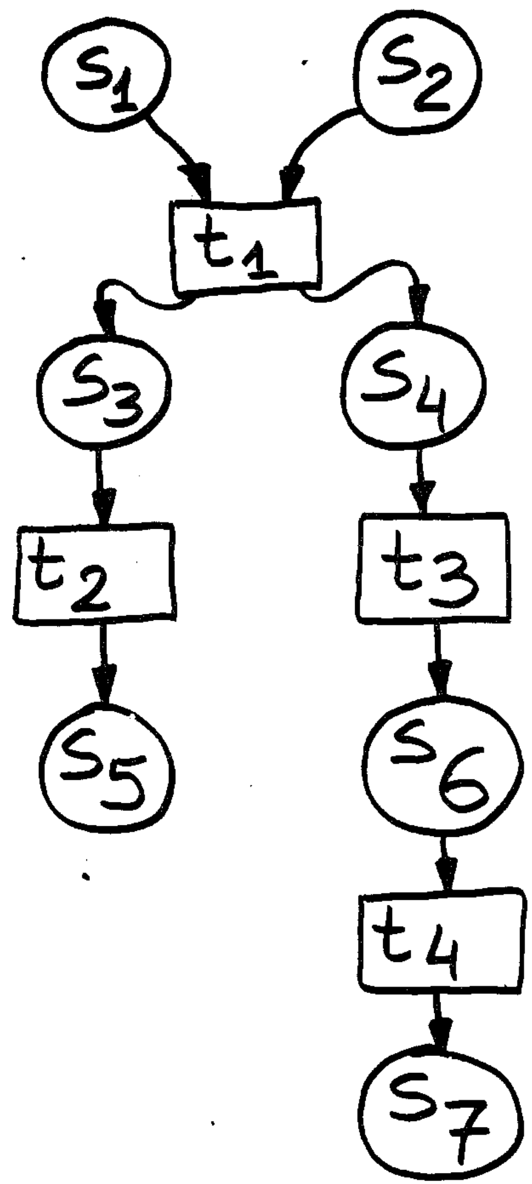
AND $\varphi(t^{\bullet}) = \varphi(t)^{\bullet}\,]$.

(iv) $\varphi({}^{0}N) \subseteq C_{in}$.

$\mathbf{P}(\mathcal{N})$



60)



62)

THEOREM

$\mathcal{N}$ EN SYSTEM

$N = (S, T, F, \varphi) \in \mathbf{P}(\mathcal{N})$

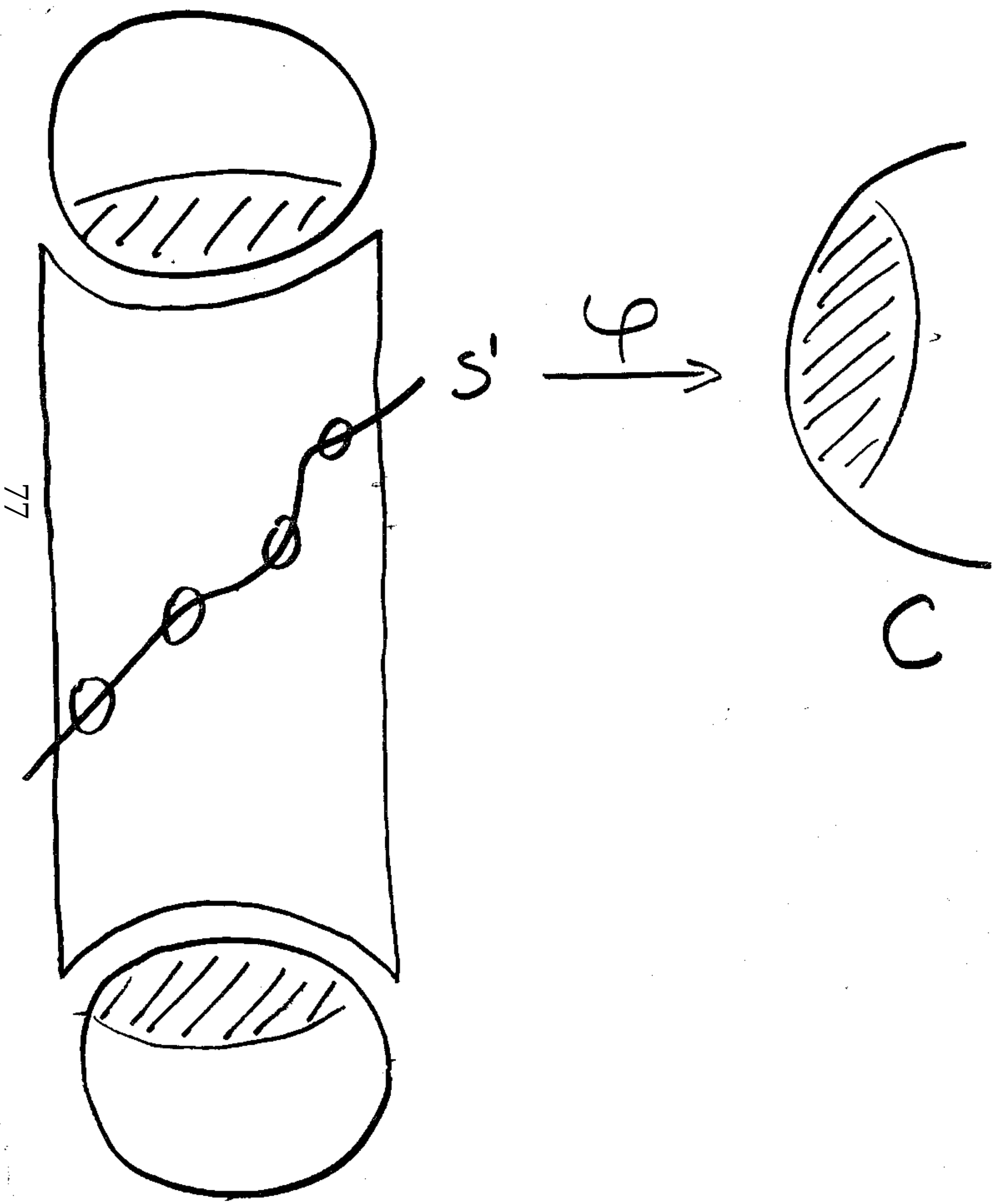
$S' \subseteq S$ A SLICE OF N

—

$(\exists C \in \mathcal{C}_{\mathcal{N}})$

$[\varphi(S') \subseteq C]$.

63)



64)

EN SYSTEM $\mathcal{N}$ is REDUCED IFF ALL EVENTS OF $\mathcal{N}$ "VISIBLE" IN $SCG(\mathcal{N})$

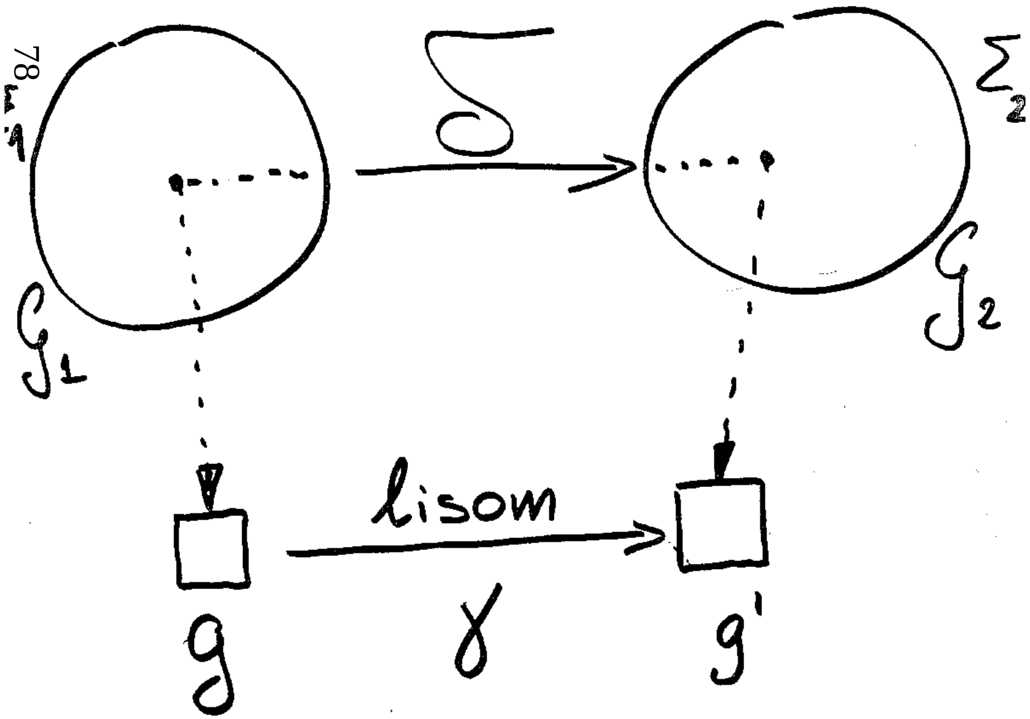
$$\left(E_{\mathcal{N}} = \bigcup_{u \in \mathcal{U}_{\mathcal{N}}} u \, . \right)$$

65)

$\mathcal{G}_1$ <u>LISOM</u> $\mathcal{G}_2$

$\exists\, \gamma : \Sigma_1 \longrightarrow \Sigma_2$

$\exists\, \delta : \mathcal{G}_1 \longrightarrow \mathcal{G}_2$



66)

EN SYSTEMS $\mathcal{N}_1, \mathcal{N}_2$

STRUCTURALLY

SIMILAR

$\mathcal{N}_1 \equiv \mathcal{N}_2$

<u>und</u>$(\mathcal{N}_1)$ <u>isom</u> <u>und</u>$(\mathcal{N}_2)$

φ

C^1_{in}, C^2_{in} RELATED

ACCORDINGLY

THEOREM

$\mathcal{N}_1, \mathcal{N}_2$ REDUCED EN'S's

$\mathbf{P}(\mathcal{N}_1)$ LISOM $\mathbf{P}(\mathcal{N}_2)$

IFF

$\mathcal{N}_1 \equiv \mathcal{N}_2$ ■

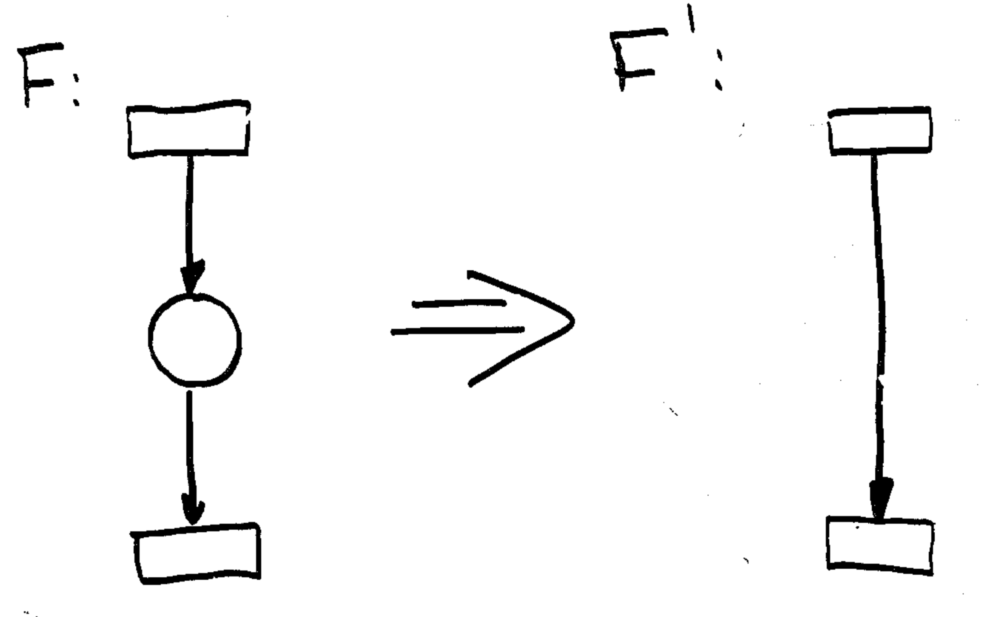
PROCESS REPRES. OF THE BEHAVIOUR OF AN EN SYSTEM IS TOO DETAILED

g A BIPARTITE GRAPH

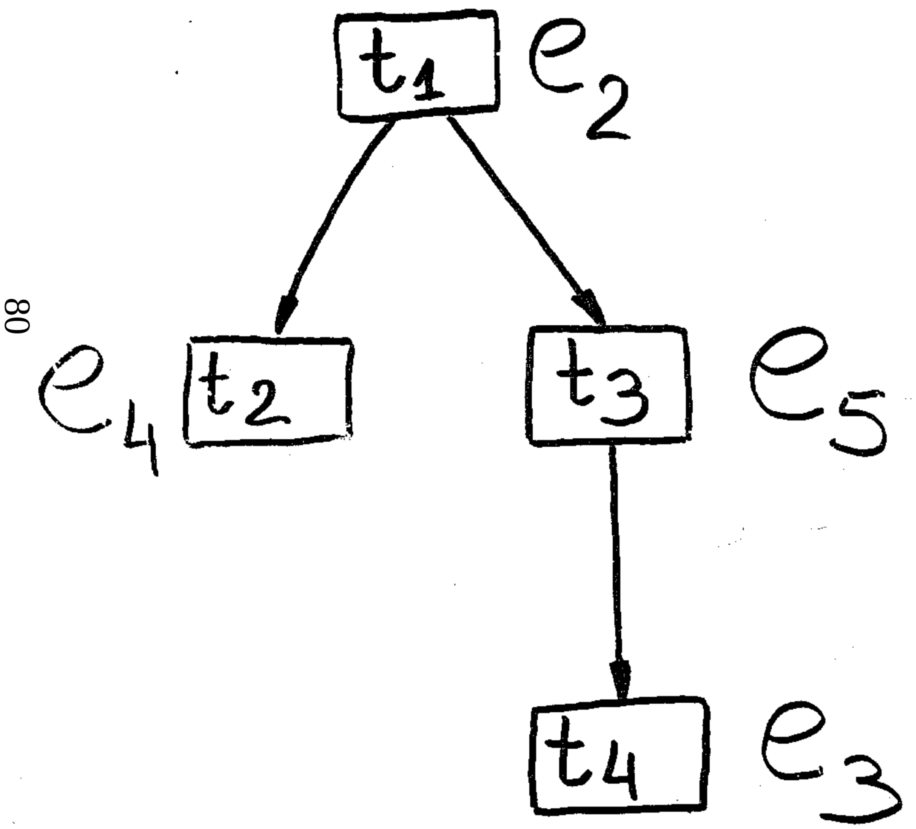
$g = (V, W, F)$

W-CONTRACTION OF g

$\underline{ctr}_W(g) = (V, F')$



69)



70)

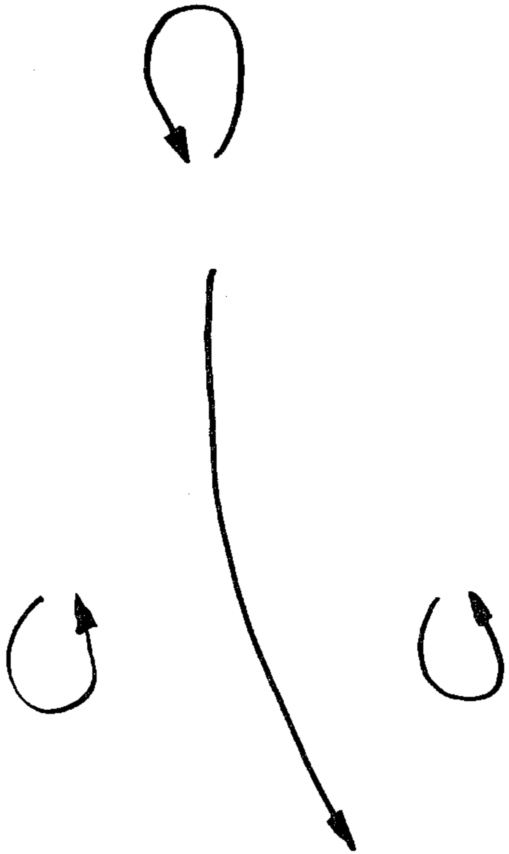
$\mathcal{N}$ AN EN SYSTEM

$N \in \mathbf{P}(\mathcal{N})$ $S = S_N$

THE S-CONTRACTED VERSION OF N IS A CONTRACTED PROCESS OF $\mathcal{N}$

$\mathbf{CP}(\mathcal{N})$

71)



72)

THEOREM

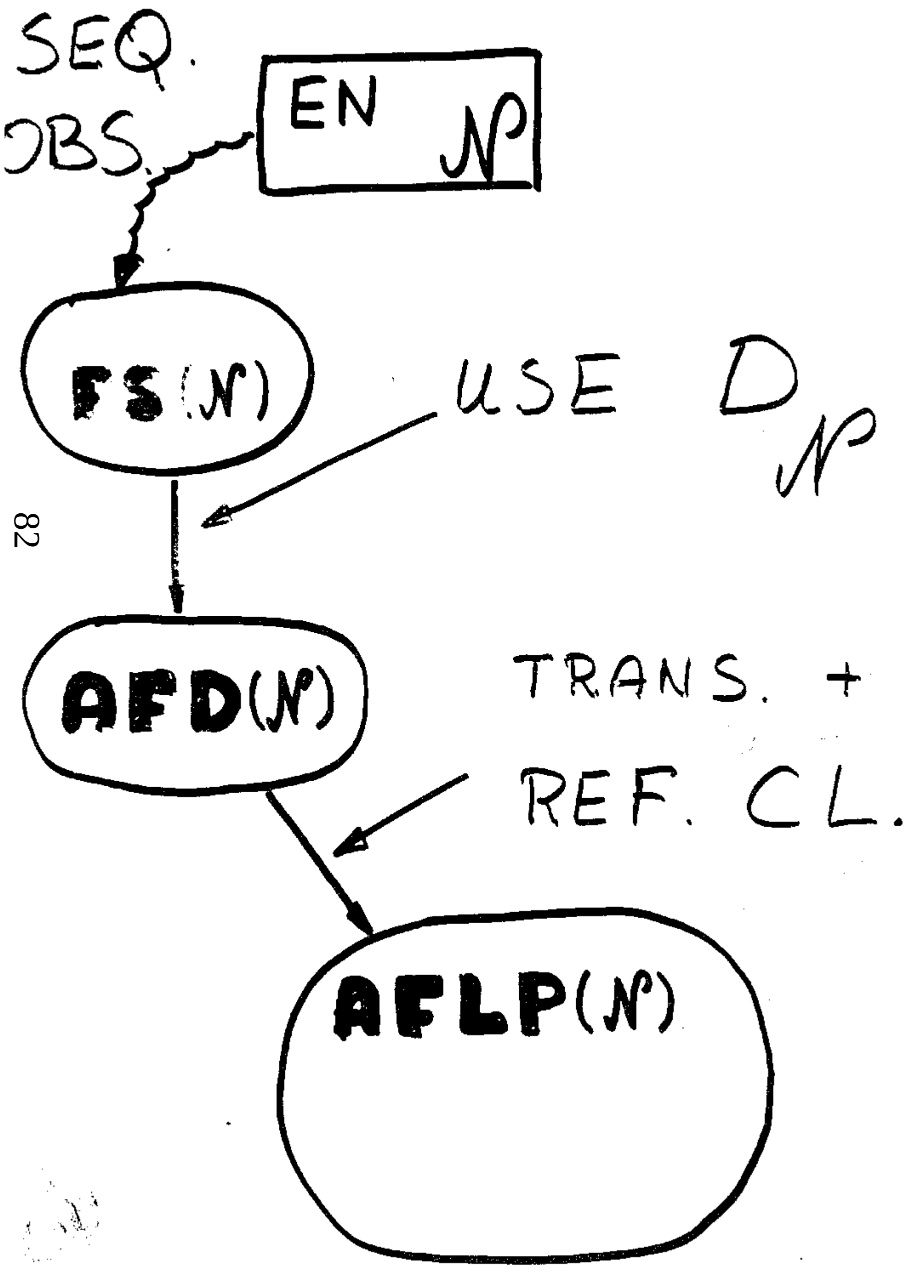
$(\exists\, \mathcal{N}_1, \mathcal{N}_2)_{\text{EN REDUCED}}$

$[\ \mathbf{CP}(\mathcal{N}_1)$ LISOM $\mathbf{CP}(\mathcal{N}_2)$

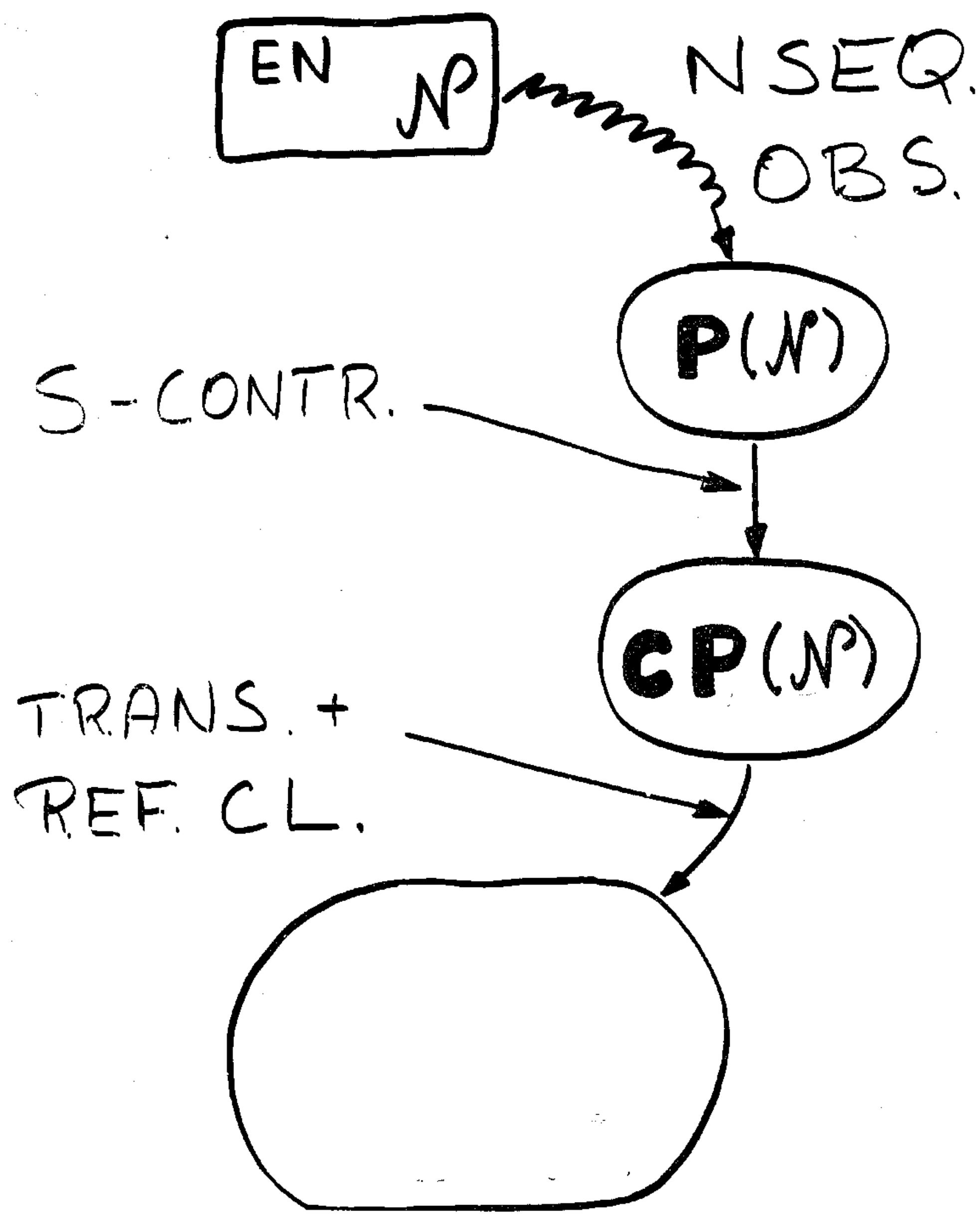
AND

$\mathcal{N}_1 \not\equiv \mathcal{N}_2\ ]$

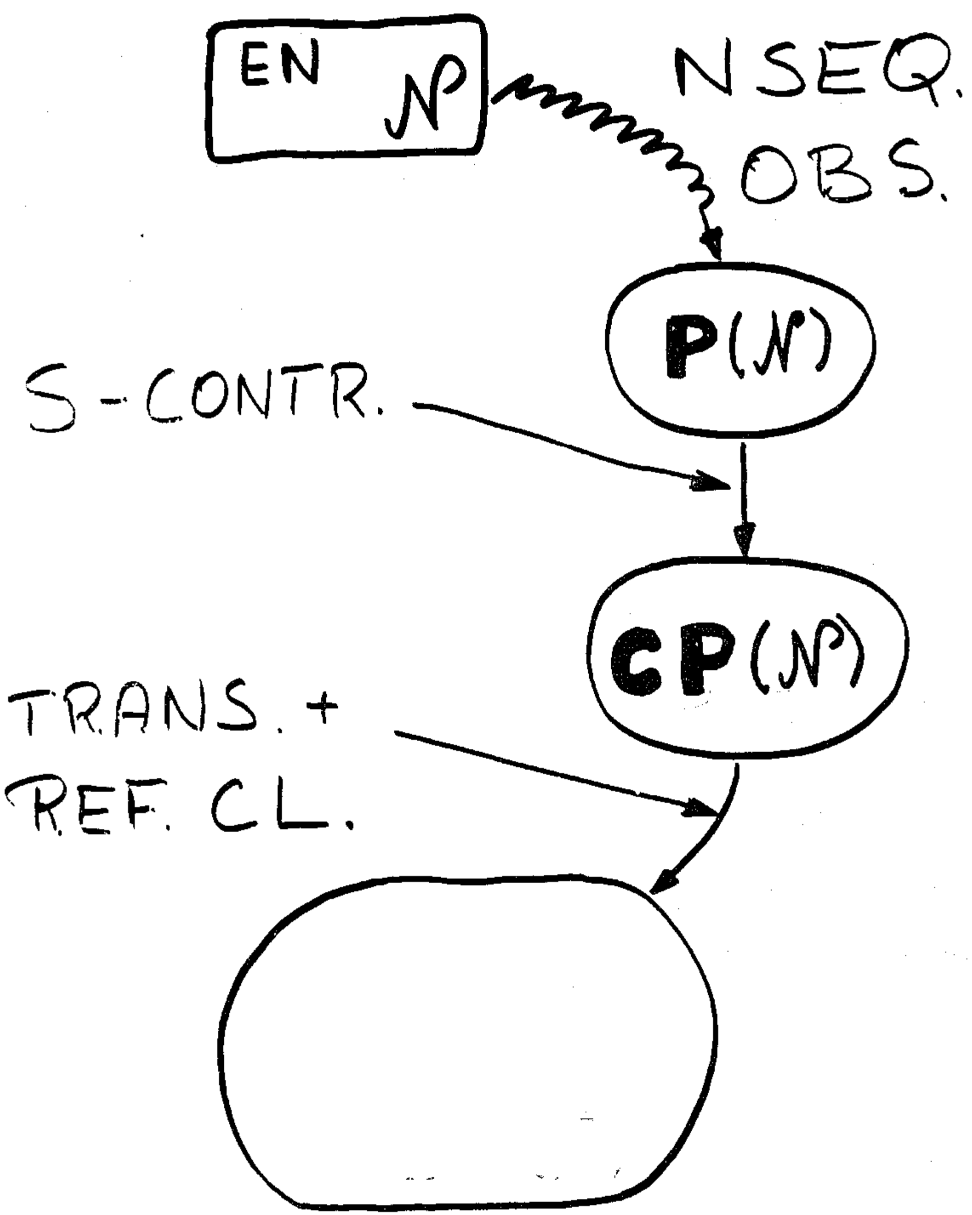
73)


SEQ.
OBS.
EN 𝒩
FS(𝒩)
USE
D
𝒩
AFD(𝒩)
TRANS. +
REF. CL.
AFLP(𝒩)

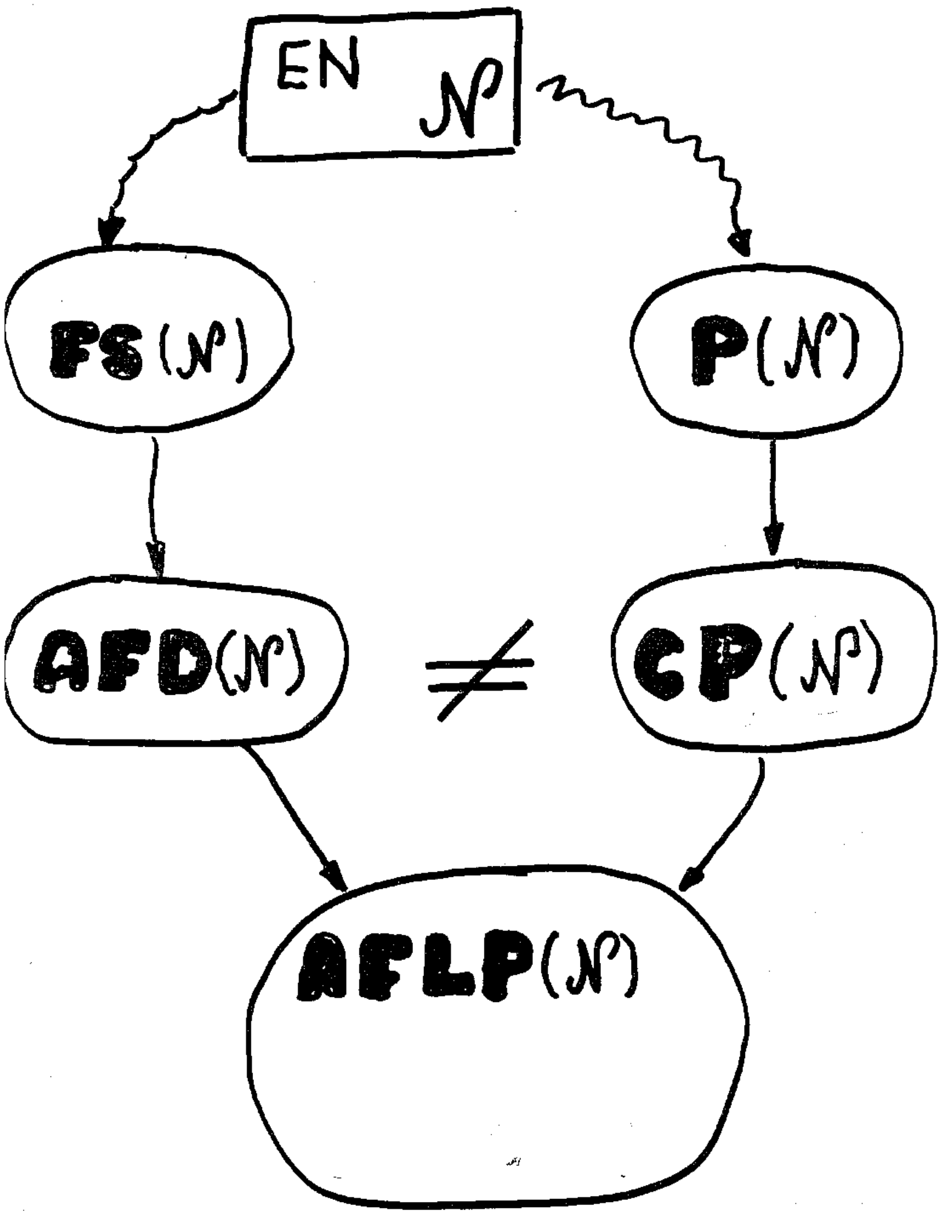
74)


EN 𝒩
NSEQ.
OBS.
P(𝒩)
S-CONTR.
CP(𝒩)
TRANS. +
REF. CL.

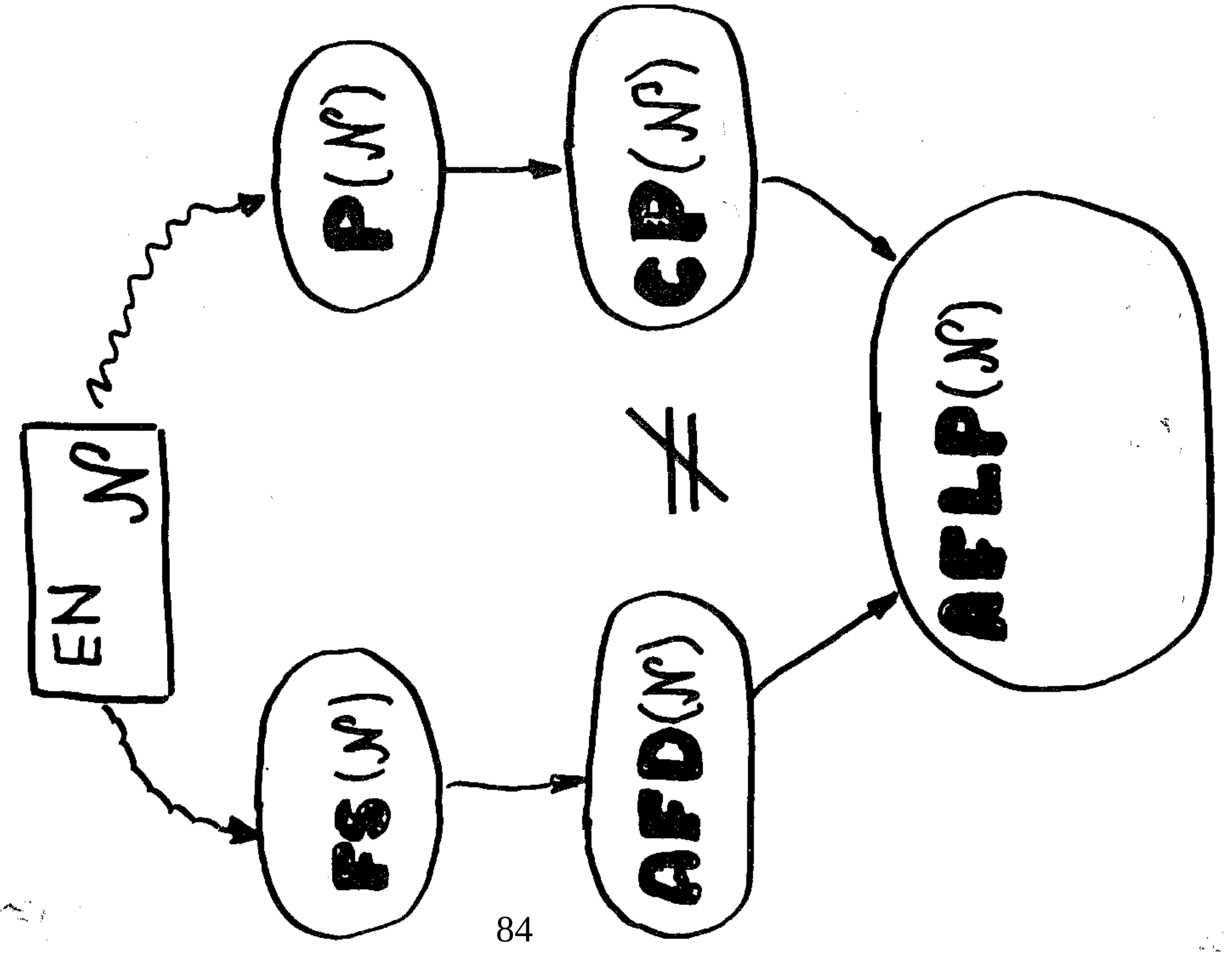
74)


EN 𝒩
NSEQ.
OBS.
P(𝒩)
S-CONTR.
CP(𝒩)
TRANS. +
REF. CL.

75)


EN 𝒩
FS(𝒩)
P(𝒩)
AFD(𝒩)
≠
CP(𝒩)
AFLP(𝒩)

75)



Introductory Tutorial

Place/Transition-Nets I

Jörg Desel, Catholic University in Eichstätt

I. **Introduction to place/transition-nets**

II. **Basic analysis techniques**

I. Introduction to place/transition nets

An example

Different features of place/transition-nets

Place/transition-nets vs en-systems

Formal definitions

Place/transition-nets

Occurrence sequences and reachability

Marking graphs

Behavioral properties

Deadlock-freedom and liveness

Boundedness and 1–safety

Reversibility

Capacities and complements

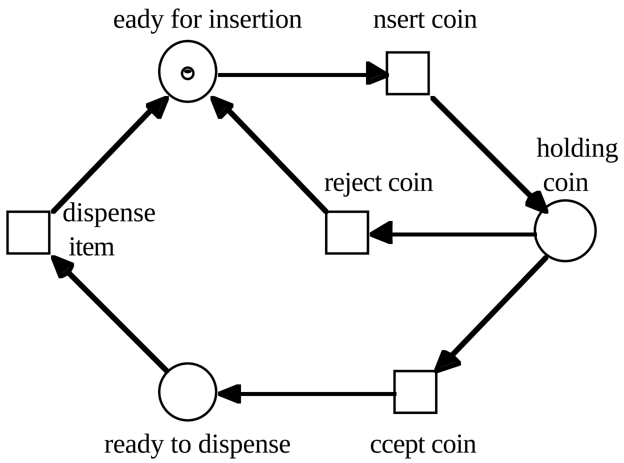
Weak and strong capacities

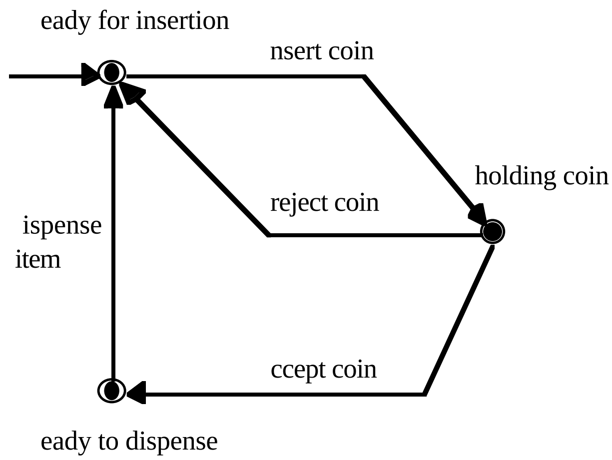
Weak and strong complements

Inhibitor arcs

An example: a vending machine

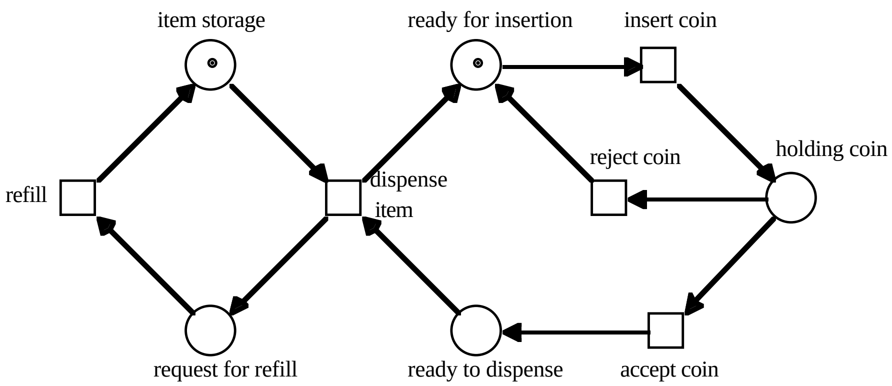
Control structure of a vending machine



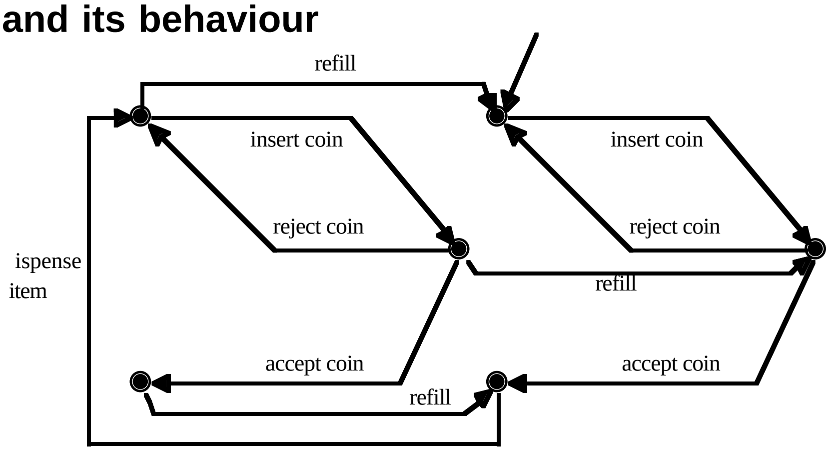


an en-system **its behaviour**

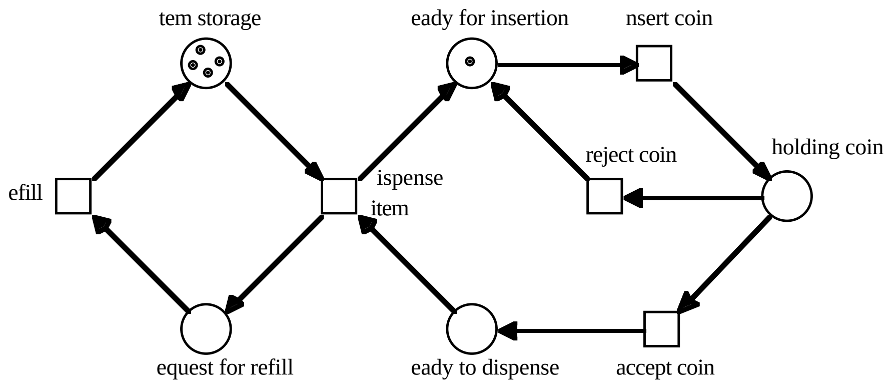
Adding concurrency: a vending machine with capacity 1 ...



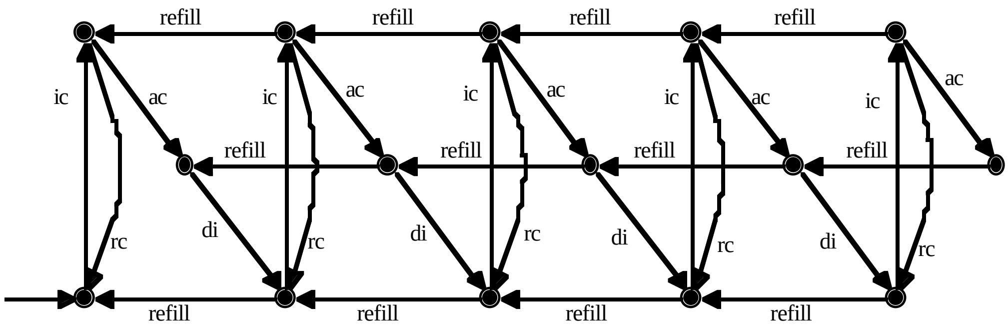
... and its behaviour



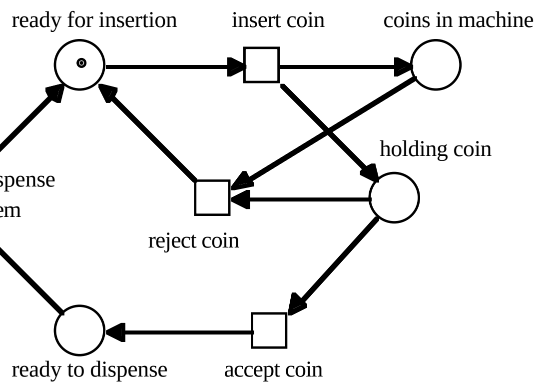
Adding bounded storages: a vending machine with capacity 4 ...



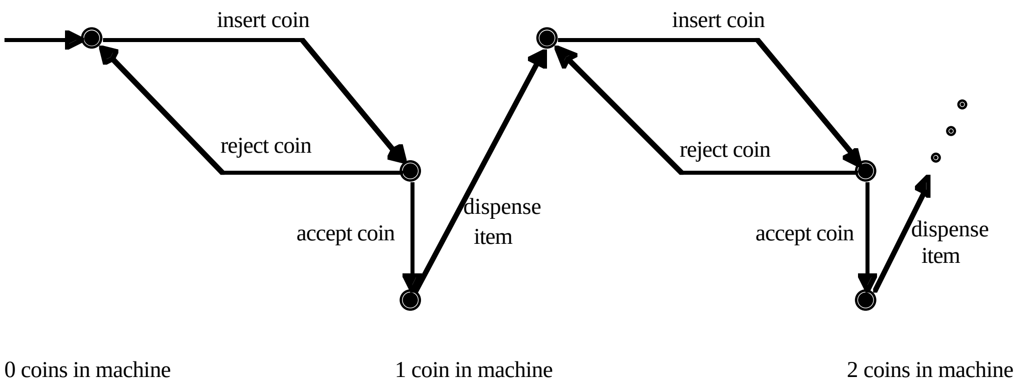
... and its behaviour



Adding unbounded counters: the control part with a counter ...

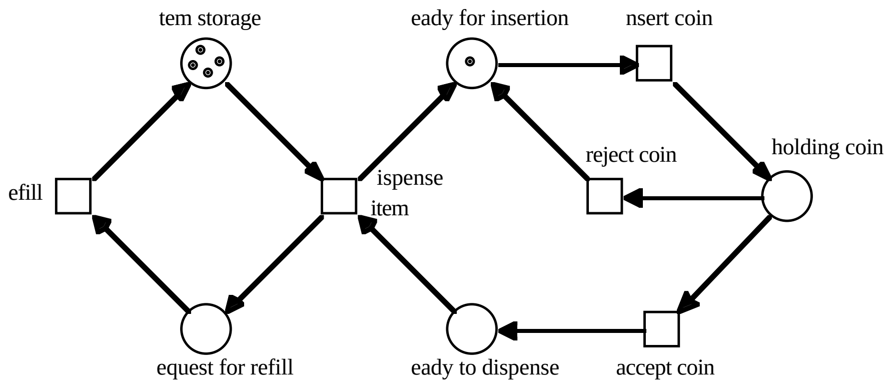


... and its behaviour

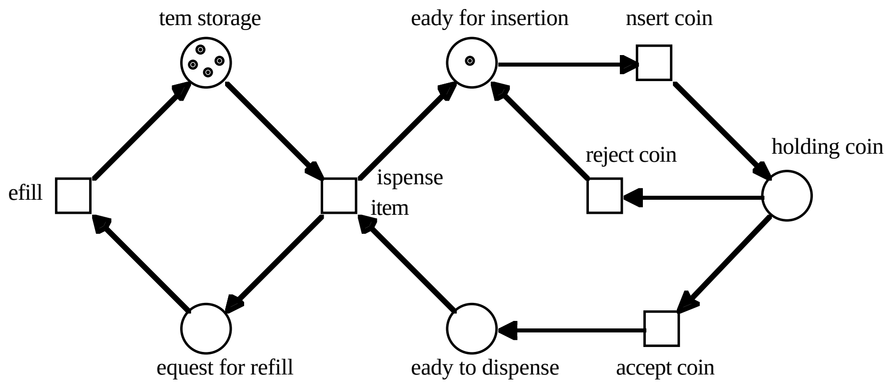


7

Adding arc weights: the vending machine selling pairs ...

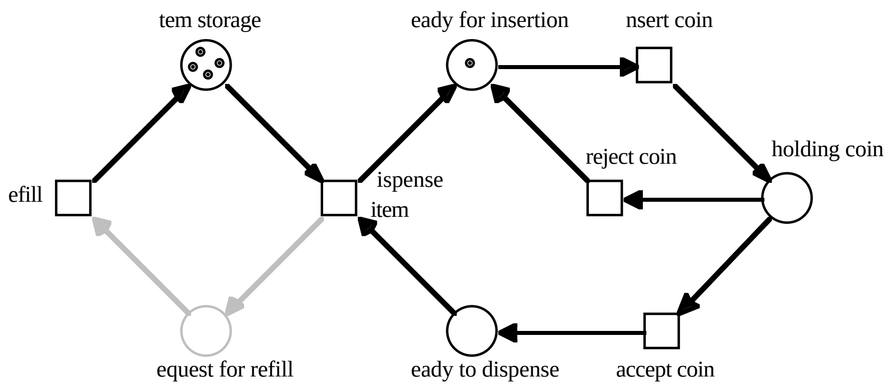


... or storing pairs

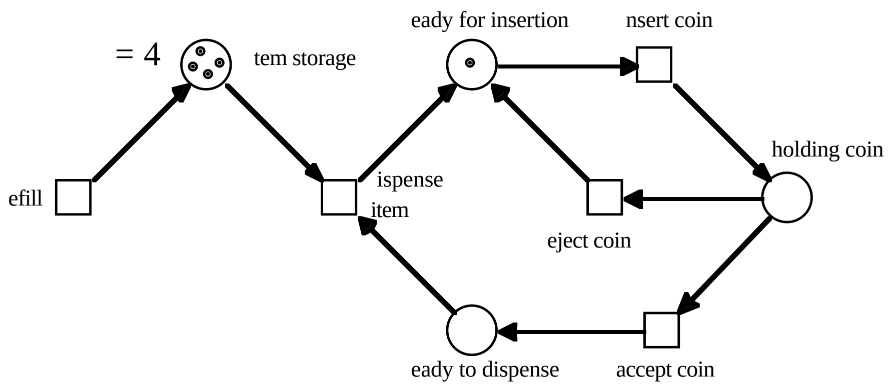


8

Adding limited capacities: replacing the place ”request for refill” ...



... by a capacity restriction



Marked place/transition-nets generalize en-systems

Each contact-free en-system is a 1-safe marked place/transition-net

Terminology:

en-system		marked p/t-net
condition	$\longrightarrow$	place
event	$\longrightarrow$	transition
case / state	$\longrightarrow$	marking
$c \subseteq$ conditions		m: places $\rightarrow \{0, 1\}$
sequential case graph	$\longrightarrow$	marking graph
		(reachability graph, state graph)

Formal definition of marked place/transition-nets

A ***marked place/transition-net (p/t-net)*** is a tuple (S, T, F, k, w, m_0) where

(S, T, F) is a net with

S – set of ***places*** (Stellen), nonempty, finite (often P is used)

T – set of ***transitions***, nonempty, finite

$F \subseteq (S \times T) \cup (T \times S)$ – ***flow relation***

$k : S \rightarrow \{1, 2, 3, \ldots\} \cup \{\infty\}$ – ***partial capacity restriction*** (default: ∞)

$w : F \rightarrow \{1, 2, 3, \ldots\}$ – ***weight function*** (default: 1)

$m_0 : S \rightarrow \{0, 1, 2, \ldots\}$ – a ***marking*** satisfying

$\forall s \in S : k(s) = \infty \ \vee \ m_0(s) \leq k(s)$

(***initial marking***)

The occurrence rule

A transition t is ***enabled*** at a marking m if

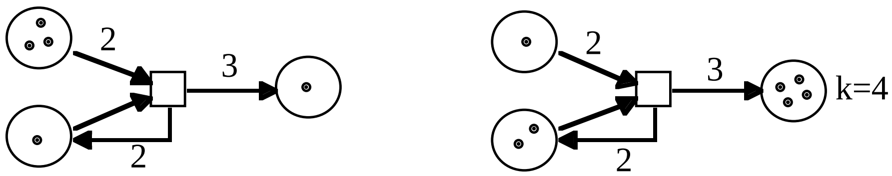
every place $s \in {}^\bullet t$ satisfies $m(s) \geq w(s,t)$ and

every place $s \in t^\bullet$ satisfies $m(s) + w(t,s) \leq k(s)$

The occurrence of t leads to the ***successor marking*** m', defined by

$$m'(s) = \begin{cases} m(s) & \text{if } s \notin {}^\bullet t \text{ and } s \notin t^\bullet \\ m(s) - w(s,t) & \text{if } s \in {}^\bullet t \text{ and } s \notin t^\bullet \\ m(s) + w(t,s) & \text{if } s \notin {}^\bullet t \text{ and } s \in t^\bullet \\ m(s) - w(s,t) + w(t,s) & \text{if } s \in {}^\bullet t \text{ and } s \in t^\bullet \end{cases}$$

Notation: $m \xrightarrow{t} m'$ $(m[t\rangle m')$



Occurrence sequences and reachability

A finite sequence $\sigma = t_1\, t_2 \ldots t_n$ of transitions is a

finite occurrence sequence leading from m_0 to m_n if

$$m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \cdots \xrightarrow{t_n} m_n$$

A marking m is ***reachable*** (from m_0) if

there is an occurrence sequence leading from m_0 to m

Notation: $[m_0\rangle$ is the set of all reachable markings

An infinite sequence $\sigma = t_1\, t_2\, t_3 \ldots$ is an

infinite occurrence sequence enabled at m_0 if

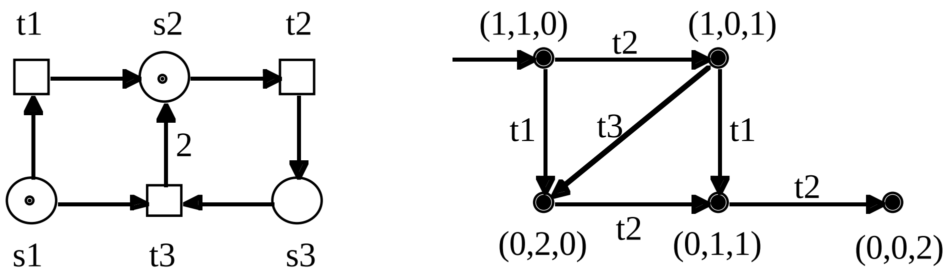
$$m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} m_2 \xrightarrow{t_3} \cdots$$

Marking graphs

The ***marking graph*** of a marked p/t-net is an edge-labeled graph with initial vertex

initial vertex	– initial marking m_0 (denoted $\rightarrow\bullet$)
vertices	– set of reachable markings $[m_0\rangle$
labeled edges	– set of triples (m, t, m') such that $m \xrightarrow{t} m'$

Example:



Lemma Each occurrence sequence corresponds to the labels of a directed path of the marking graph starting with the initial vertex, and vice versa.

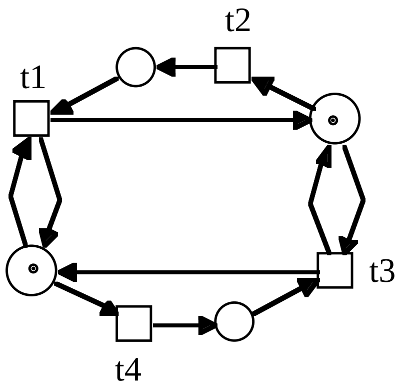
Behavioral properties of marked p/t-nets

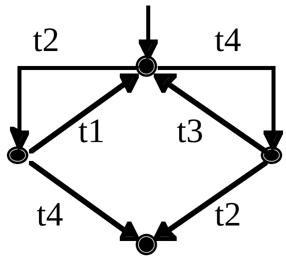
A marked p/t-net is

terminating	– if there is no infinite occurrence sequence
deadlock-free	– if each reachable marking enables a transition
live	– if each reachable marking enables an occurrence sequence containing all transitions
bounded	– if, for each place s, there is a bound $b(s)$ such that $m(s) \leq b(s)$ for every reachable marking m
1–safe	– if $b(s)$ = 1 is a bound for each place s
reversible	– if m_0 is reachable from each other reachable marking

Example The vending machines are deadlock-free and live.
Some are 1–safe, some are bounded, some are unbounded.
The bounded vending machines are reversible.

A marked p/t-net which is not deadlock-free and its marking graph

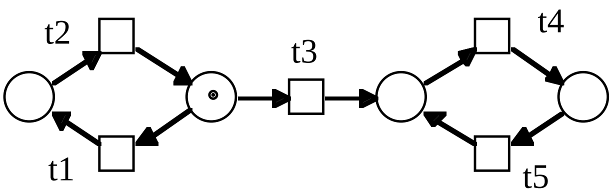


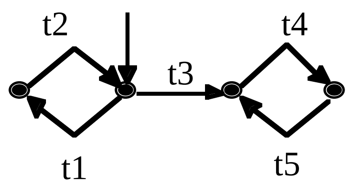


Proposition A marked p/t-net is deadlock-free if and only if its marking graph has no vertex without successor

Proposition No deadlock-free marked p/t-net is terminating (but the converse does not necessarily hold)

A deadlock-free marked p/t-net which is not live

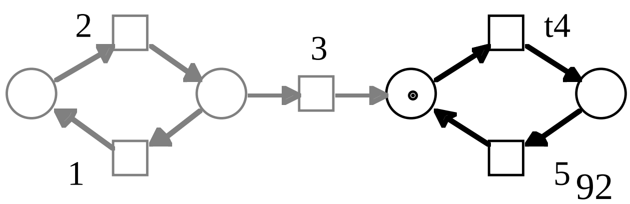


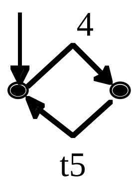


Proposition Every live marked p/t-net is deadlock-free (this does not hold for nets without transitions)

Proposition A marked p/t-net is live if and only if at no reachable marking a transition is dead (cannot become enabled again)

Example Some transitions are dead at a reachable marking





Proposition A marked p/t-net is bounded if and only if its set of reachable markings is finite (its marking graph is finite)

Proof

($\Leftarrow$) The maximal number of tokens on a place can be taken as its bound.

($\Rightarrow$) If a place s is bounded by $b(s)$ then it can be in at most $b(s)+1$ different states, vic.

$$m(s) = 0, m(s) = 1, \ldots, m(s) = b(s).$$

So the number of reachable markings does not exceed

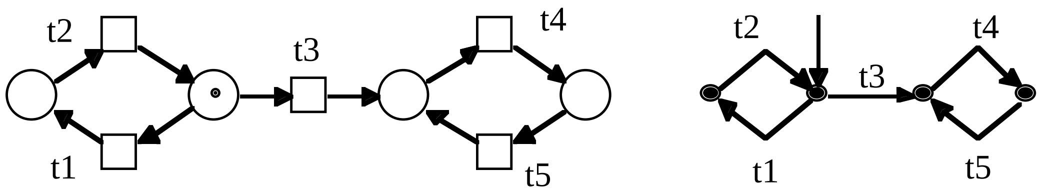
$$(b(s_1) + 1) \cdot (b(s_2) + 1) \cdots (b(s_n) + 1)$$

where $\{s_1, s_2, \ldots, s_n\}$ is the (finite !) set of places

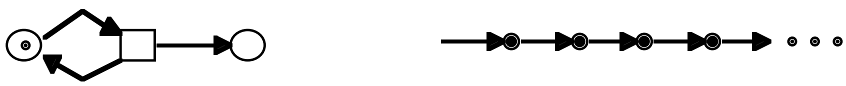
Corollary A 1–safe marked p/t-net with n places has at most 2^n reachable markings

Proposition A marked p/t-net is reversible if and only if its marking graph is strongly connected

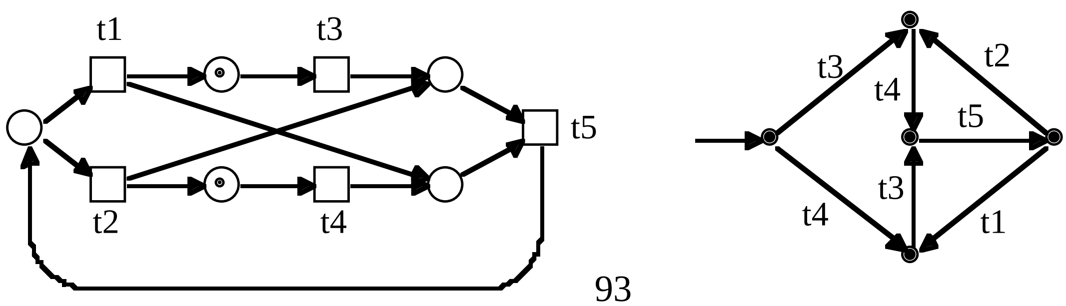
Example a 1–safe non-live marked p/t-net which is not reversible



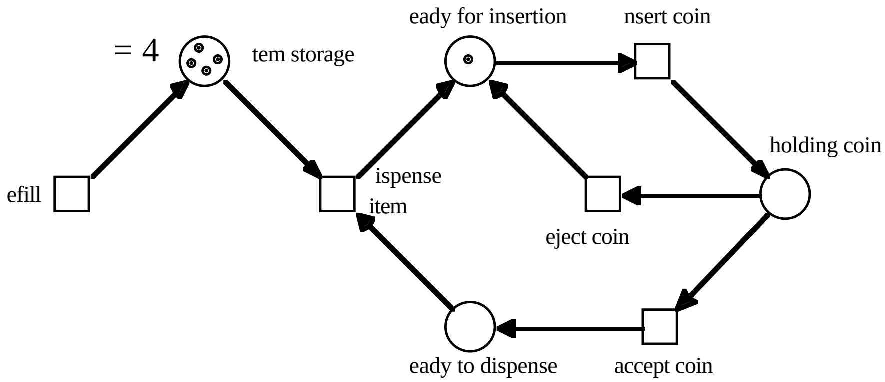
Example an unbounded marked p/t-net which is not reversible



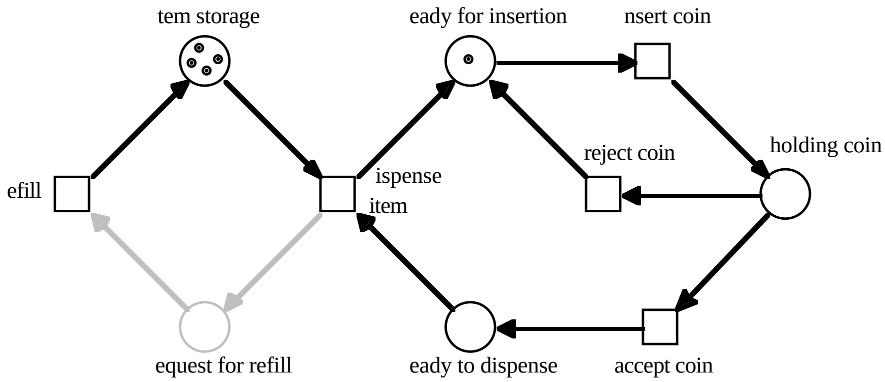
Example a live and 1–safe marked p/t-net which is not reversible



Substituting capacities ...



... by complement places



Weak capacities

. . . guarantee bounds of places

weak enabling condition:

a transition t is enabled at a marking m if

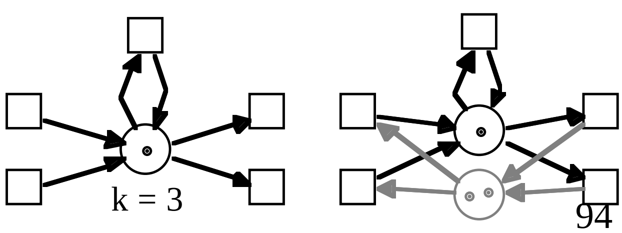
every place $s \in {}^{\bullet}t$ satisfies $m(s) \geq w(s,t)$ and

every place $s \in t^{\bullet} \setminus {}^{\bullet}t$ satisfies $m(s) + w(t,s) \leq k(s)$ and

every place $s \in t^{\bullet} \cap {}^{\bullet}t$ satisfies $m(s) - w(s,t) + w(t,s) \leq k(s)$

Proposition If $k(s)$ is finite then s is $k(s)$-bounded

Replacing a weak capacity restrictiction by a *weak complement*



Strong capacities

. . . generalize contact of en-systems

strong enabling condition:

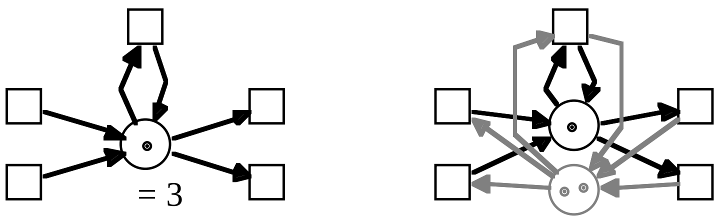
a transition t is enabled at a marking m if

every place $s \in {}^\bullet t$ satisfies $m(s) \geq w(s,t)$ and

every place $s \in t^\bullet$ satisfies $m(s) + w(t,s) \leq k(s)$

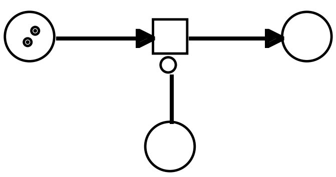
Proposition each en-system is equivalent to a marked p/t-net without arc weights and with the strong capacity restriction $k(s) = 1$ for every place s

Replacing a strong capacity restrictiction by a *strong complement*

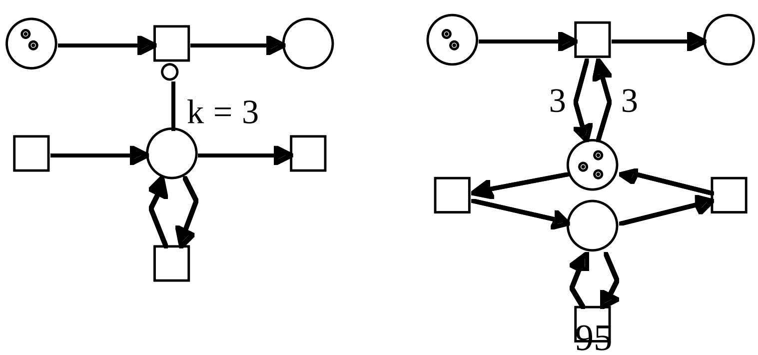


Inhibitor arcs for null tests

inhibitor enabling condition: If (s,t) is an inhibitor arc then t is only enabled at a marking m if $m(s) = 0$



Replacing an inhibitor arc at a bounded place by a weak complement



II. Basic analysis techniques

Linear-algebraic techniques

- The marking equation
- Place invariants
- Transition invariants

Structural techniques

- Siphons
- Traps
- The siphon/trap property

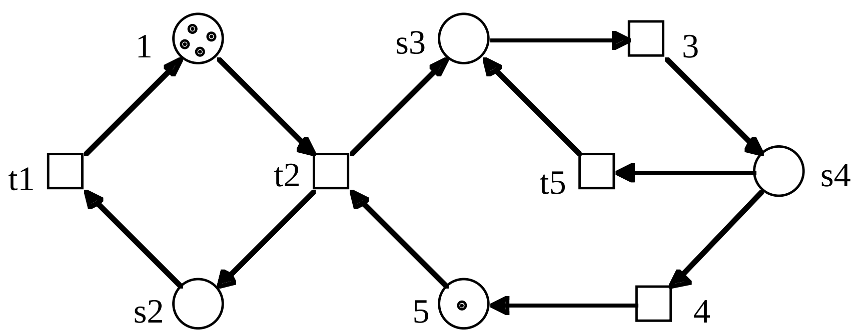
Restricted net classes

- State machines
- Marked graphs
- Free-choice nets

Causal Semantics

- Occurrence nets
- Process nets

Linear-algebraic representation of markings and transitions

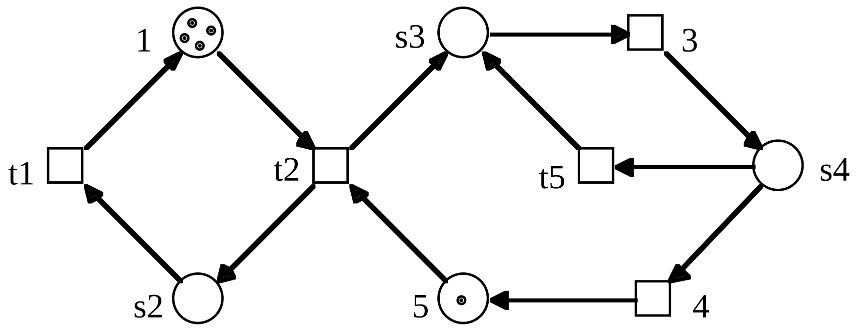


vector representation of the marking m_0: $\vec{m_0}$ = $(4, 0, 0, 0, 1)$

vector representation of the transition t_2: $\vec{t_2}$ = $(-1, 1, 1, 0, -1)$

$$m_0 \xrightarrow{t_2} m_1 \quad \Rightarrow \quad \vec{m_0} + \vec{t_2} = \vec{m_1} \;=\; (3, 1, 1, 0, 0)$$

Matrix representation of a net



incidence matrix of the net:

(N) =

	$\vec{t_1}$	$\vec{t_2}$	$\vec{t_3}$	$\vec{t_4}$	$\vec{t_5}$
$\vec{s_1}$	1	−1	0	0	0
$\vec{s_2}$	−1	1	0	0	0
$\vec{s_3}$	0	1	−1	0	1
$\vec{s_4}$	0	0	1	−1	−1
$\vec{s_5}$	0	−1	0	1	0

The marking equation

$$m_0 \xrightarrow{t_2\ t_3\ t_5\ t_1\ t_3} m \quad \Rightarrow \quad \vec{m_0} + \vec{t_2} + \vec{t_3} + \vec{t_5} + \vec{t_1} + \vec{t_3} = \vec{m}$$

$$\vec{m_0} + (1 \cdot \vec{t_1}) + (1 \cdot \vec{t_2}) + (2 \cdot \vec{t_3}) + (0 \cdot \vec{t_4}) + (1 \cdot \vec{t_5}) = \vec{m}$$

$$\vec{m_0} + (N) \cdot \underbrace{(1, 1, 2, 0, 1)}_{\textbf{Parikh vector} \text{ of } t_2 t_3 t_5 t_1 t_3} = \vec{m}$$

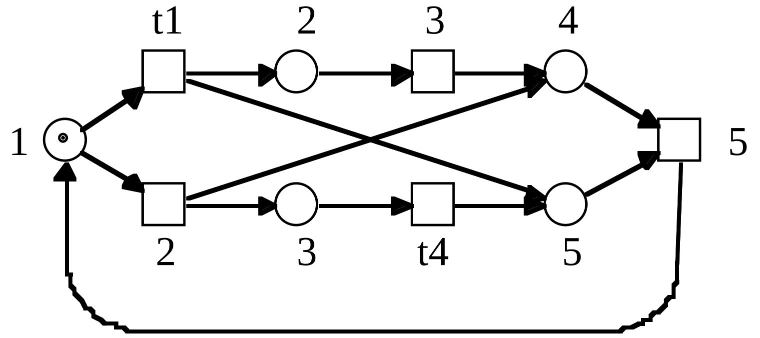
The Marking Equation If $m_0 \xrightarrow{\sigma} m$ and $\mathcal{P}(\sigma)$ denotes the Parikh vector of σ then

$$\vec{m_0} + (N) \cdot \mathcal{P}(\sigma) = \vec{m}$$

. . . yields a necessary condition for reachability of a marking:

A marking m is only reachable from m_0 if $\vec{m_0} + (N) \cdot \vec{x} = \vec{m}$ has a solution for $\vec{x}$ in $\mathbb{N}^*$.

Example: a live and 1–safe marked p/t-net

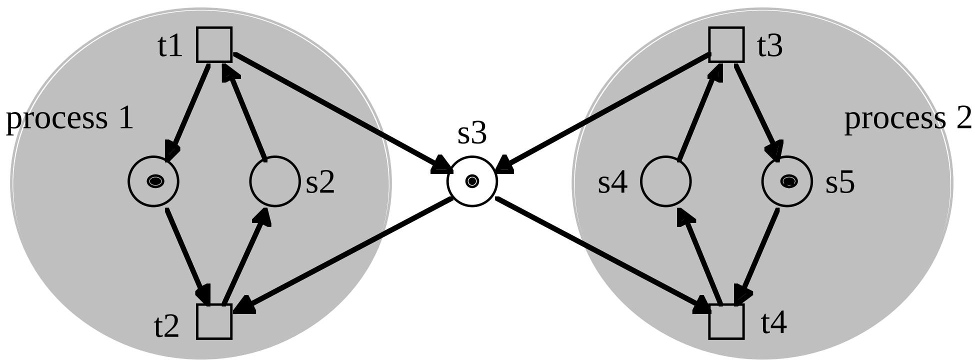


reachable markings	solutions to the marking equation
$(1, 0, 0, 0, 0)$	$(0, 0, 0, 0, 0),\ (1, 0, 1, 0, 1),\ (0, 1, 0, 1, 1), \ldots$
$(0, 1, 0, 0, 1)$	$(1, 0, 0, 0, 0), \ldots$
$(0, 0, 1, 1, 0)$	$(0, 1, 0, 0, 0), \ldots$
$(0, 0, 0, 1, 1)$	$(1, 0, 1, 0, 0),\ (0, 1, 0, 1, 0), \ldots$
non-reachable marking	**solutions to the marking equation**
$(0, 1, 1, 0, 0)$	$(1, 1, 0, 0, 1) \ldots$

Consequence: solubility of the marking equation is not sufficient for reachability

Place invariants

Example: mutual exclusion



Every reachable marking m satisfies $m(s_2) + m(s_4) \leq 1$

1) $m(s_2) + m(s_3) + m(s_4) = 1$ holds initially

2) $m(s_2) + m(s_3) + m(s_4) = 1$ is stable $\rightarrow$ will be shown by a ***place invariant***

3) $m(s_2) + m(s_3) + m(s_4) = 1 \;\Rightarrow\; m(s_2) + m(s_4) \leq 1$

Place invariants

Three equivalent definitions:

A ***place invariant*** of a net N is a vector $\vec{i}$ satisfying

(1) $\sum_{s \in {}^\bullet t} \vec{i}_s = \sum_{s \in t^\bullet} \vec{i}_s$ for every transition t of N

(2) $\vec{i} \cdot \vec{t} = 0$ for every transition t of N

(3) $\vec{i} \cdot [N] = (0, 0, \ldots, 0)$

The token conservation law for a place invariant $\vec{i}$:

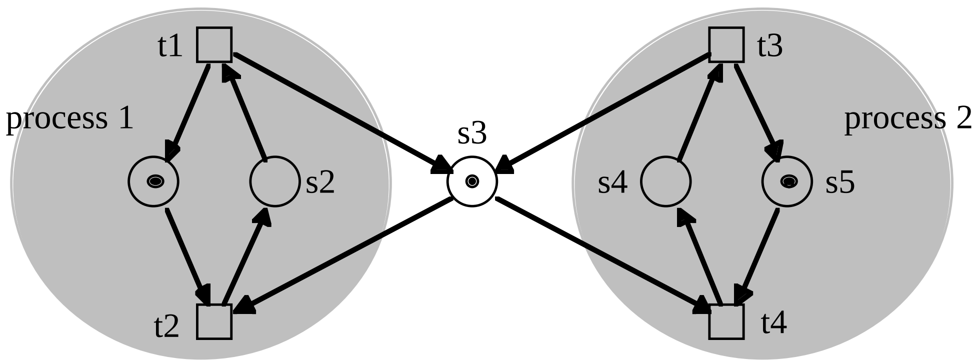
If m is reachable from m_0 then $\vec{i} \cdot \vec{m_0} = \vec{i} \cdot \vec{m}$

Proof: $m_0 \xrightarrow{\sigma} m \;\Rightarrow\; \vec{m_0} + [N] \cdot \mathcal{P}[\sigma] = \vec{m}$

$$\Rightarrow\; \vec{i} \cdot \vec{m_0} + \underbrace{\vec{i} \cdot [N]}_{=(0,\ldots,0)} \cdot \mathcal{P}[\sigma] = \vec{i} \cdot \vec{m}$$

$$\Rightarrow\; \vec{i} \cdot \vec{m_0} = \vec{i} \cdot \vec{m}$$

Proving stability



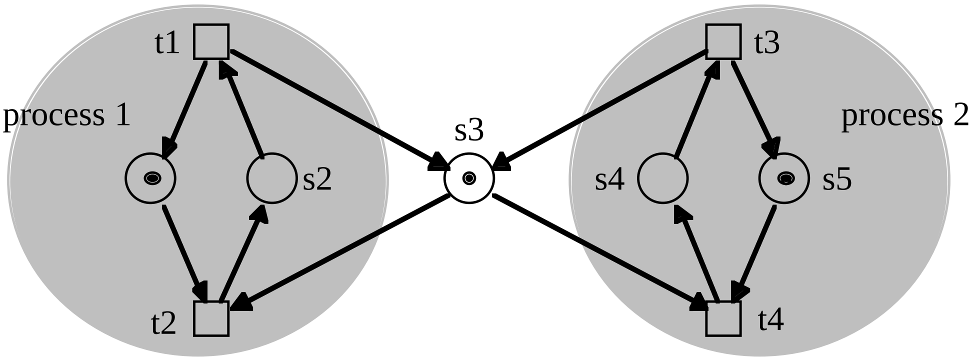
The number of tokens on $\{s_2, s_3, s_4\}$ is not changed by transition occurrences.

$\Rightarrow \vec{i} = (0, 1, 1, 1, 0)$ is a place invariant.

$\vec{i} \cdot \vec{m_0} = 1$ implies $\vec{i} \cdot \vec{m} = 1$ for each reachable marking M.

$\Rightarrow m(s_2) + m(s_3) + m(s_4) = 1$ is stable.

Further place invariants



(0, 1, 1, 1, 0) mutual exclusion

(0, 1, 1, 0, −1) $m(s_2) + m(s_3) = m(s_5)$

if s_2 is marked then s_5 is marked

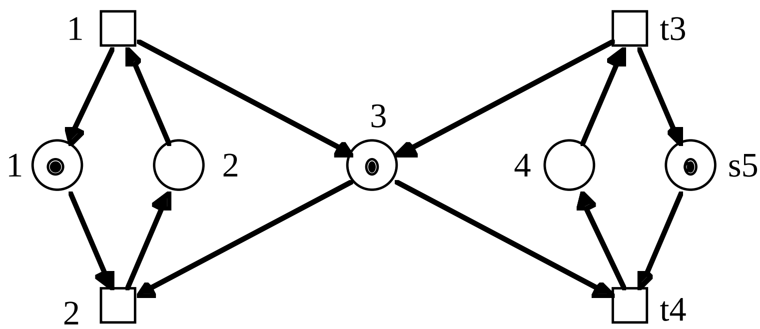
(1, 1, 0, 0, 0) $m(s_1) + m(s_2) = 1$

$m(s_1), m(s_2) \leq 1$, the places s_1 and s_2 are bounded

A necessary condition for liveness

Proposition: In a live marked p/t-net without isolated places, each place invariant $\vec{i}$ without negative entries and with some positive entry $\vec{i}_s$ satisfies $\vec{i} \cdot \vec{m_0} > 0$.

Proof: otherwise transitions in ${}^\bullet s \cup s^\bullet$ are dead.



Examples: place invariants (1, 1, 0, 0, 0), (0, 0, 0, 1, 1), (0, 1, 1, 1, 0)

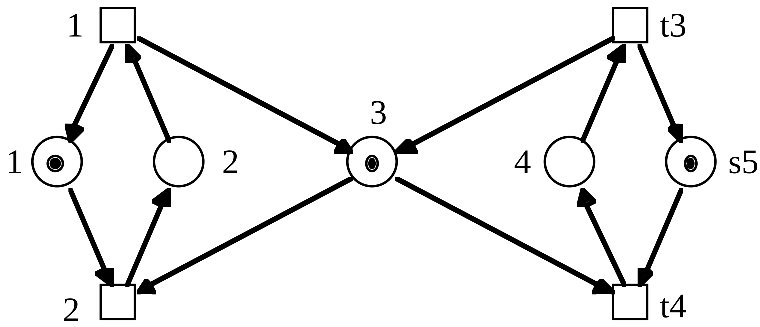
A sufficient condition for boundedness:

Proposition: Each marked p/t-net with a place invariant $\vec{i}$ satisfying $\vec{i}_s > 0$ for each place s is bounded.

Proof: m is reachable $\Rightarrow$ $\vec{i} \cdot \vec{m} = \vec{i} \cdot \vec{m_0}$.

$\Rightarrow$ $\vec{i}_s \cdot \vec{m}_s \leq \vec{i} \cdot \vec{m} = \vec{i} \cdot \vec{m_0}$.

$$\Rightarrow \quad m(s) = \vec{m}_s \leq \frac{\vec{i} \cdot \vec{m_0}}{\vec{i}_s}$$

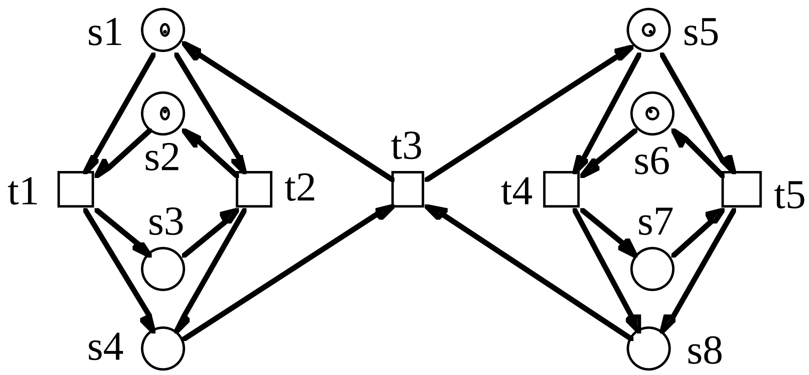


Example: place invariant $(1, 2, 1, 2, 1)$

Place invariants and the marking equation

Proposition There is a place invariant $\vec{i}$ satisfying $\vec{i} \cdot \vec{m_0} \neq \vec{i} \cdot \vec{m}$ if and only if $\vec{m_0} + [N] \cdot \vec{x} = \vec{m}$ has no rational-valued solution for $\vec{x}$.

Example:



$\vec{m_0} + [N] \cdot (1, 0, 1, \frac{1}{2}, \frac{1}{2}) = \vec{m} = (1, 0, 1, 0, 1, 1, 0, 0)$

$\Rightarrow$ no place invariant proves the non-reachability of m.

But the marking equation has no solution in $\mathbb{N}^*$

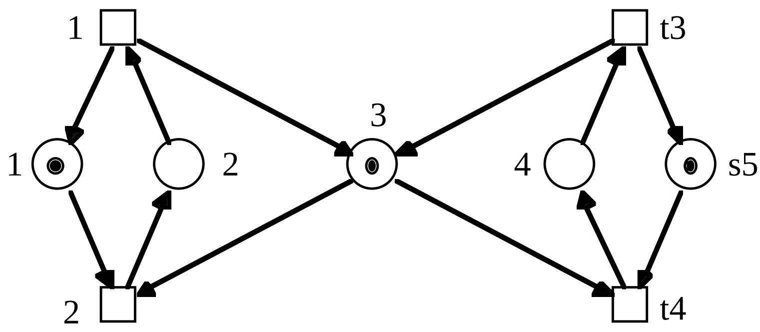
$\rightarrow$ ***modulo place invariants***

Transition invariants

A ***transition invariant*** of a net N is a vector $\vec{j}$ satisfying

$$[N] \cdot \vec{j} = (0, 0, \ldots, 0)$$

Example:



Transition invariants: $(1, 1, 0, 0)$, $(0, 0, 1, 1)$, $(2, 2, 1, 1)$

Proposition Let $m_0 \xrightarrow{\sigma} m$ be an occurrence sequence.

$m_0 = m$ if and only if $\mathcal{P}[\sigma]$ is a transition invariant

Proof: follows immediately from $\vec{m_0} + [N] \cdot \mathcal{P}[\sigma] = \vec{m}$

A necessary condition for liveness and boundedness

Proposition: Each live and bounded marked p/t-net has a transition invariant $\vec{j}$ satisfying $\vec{j}(t) > 0$ for each transition t.

Proof: By liveness, there exist occurrence sequences

$$m_0 \xrightarrow{\sigma_1} m_1 \xrightarrow{\sigma_2} m_2 \xrightarrow{\sigma_3} \cdots$$

such that all transitions occur in every σ_i.

By boundedness, $m_i = m_j$ for some $i < j$.

$\Rightarrow m_i \xrightarrow{\sigma_{i+1}} \cdots \xrightarrow{\sigma_j} m_j = m_i$.

$\Rightarrow \vec{j} = \mathcal{P}[\sigma_{i+1} \cdots \sigma_j]$ is a suitable transition invariant.

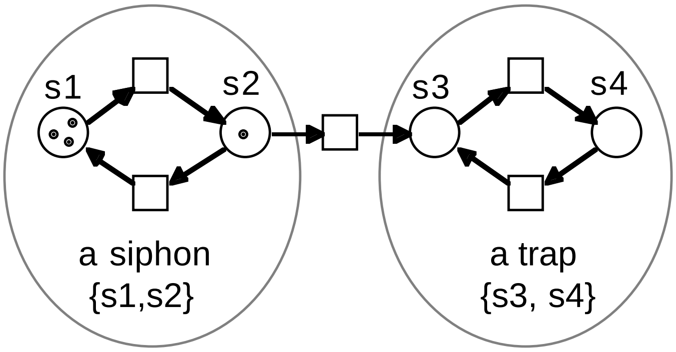
Structural Techniques

A siphon is a set of places which, once unmarked, never gains a token again

S is a ***siphon*** if ${}^\bullet S \subseteq S^\bullet$, i.e. if $t^\bullet \cap S \neq \emptyset$ implies ${}^\bullet t \cap S \neq \emptyset$.

A trap is a set of places which, once marked, never looses all tokens

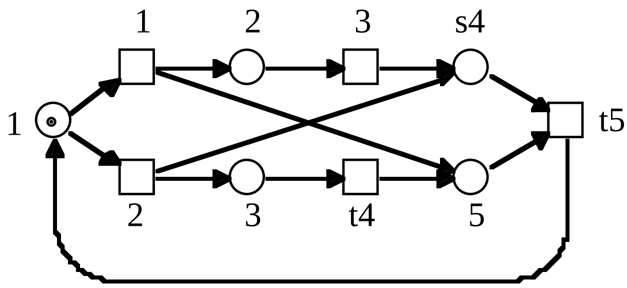
S is a ***trap*** if $S^\bullet \subseteq {}^\bullet S$, i.e. if ${}^\bullet t \cap S \neq \emptyset$ implies $t^\bullet \cap S \neq \emptyset$.



If a marking m satisfies $m(s_1)$ = $m(s_2)$ = 0 then so do all follower markings.

If a marking m satisfies $m(s_3) + m(s_4) > 0$ then so do all follower markings.

Example for the use of a trap



$\{s_1, s_4, s_5\}$ is an initially marked trap

$\Rightarrow$ the marking $(0, 1, 1, 0, 0)$ is not reachable.

Siphons and traps, liveness and deadlock-freedom

Proposition: In a live marked p/t-net without isolated places,
each nonempty siphon contains an initially marked place

Proof: otherwise, for each place s of the siphon, all transitions in ${}^\bullet s \cup s^\bullet$ are dead

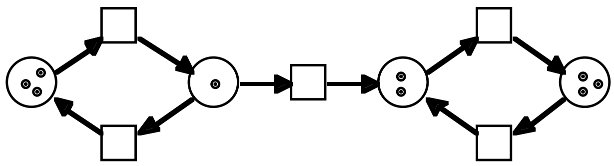
Proposition: Assume a marked p/t-net with some transition,
without capacity restrictions and arc weights.
If each nonempty siphon includes an initially marked trap
then the marked p/t-net is deadlock-free

Proof: the set of unmarked places at a dead marking is a nonempty siphon.
This siphon contains no marked trap.
$\Rightarrow$ It contains no initially marked trap.

Restricted net classes

State machines are marked p/t-nets without branched transitions,
i.e. $|{}^\bullet t| = |t^\bullet| = 1$ for each transition,
without arc weights and
without capacity restrictions.

Example



Proposition Each marked state machine is bounded.

Proposition A marked state machine is live if and only if
it is strongly connected and some place is initially marked.

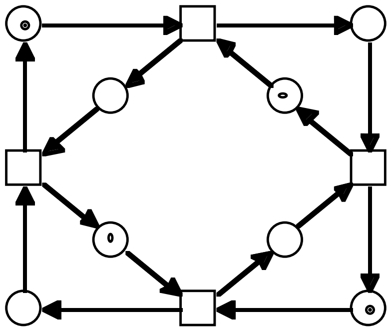
Marked graphs are marked p/t-nets without branched places,

i.e. $|{}^\bullet s| = |s^\bullet| = 1$ for each place,

without arc weights and

without capacity restrictions.

Example

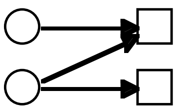
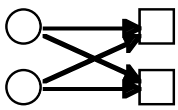


Proposition A marked graph is live if and only if each cycle carries a token initially.

Proposition It is moreover 1–safe if and only if each place belongs to a cycle with exactly one token.

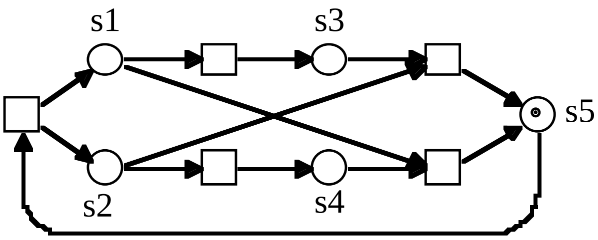
Free-choice nets are marked p/t-nets without arc weights and capacity restrictions satisfying

$(s, t) \in F \;\Rightarrow\; {}^\bullet t \times s^\bullet \subseteq F$ for each place s and transition t

If  then 

Proposition A free-choice net without isolated places is live if and only if each nonempty siphon includes an initially marked trap.

Example

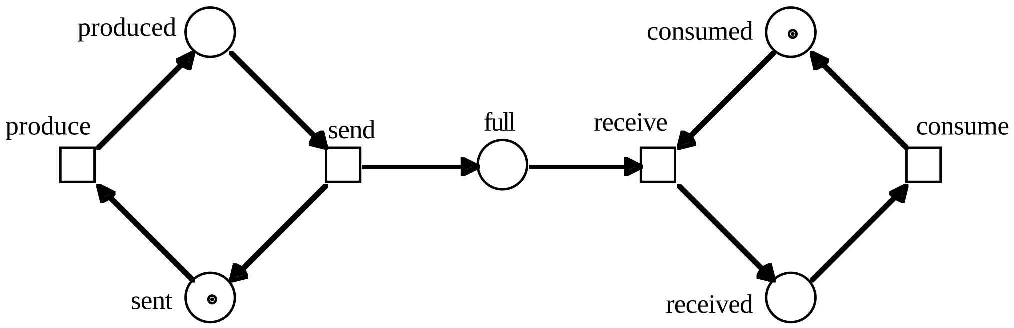


$\{s_1, s_2, s_5\}$ is a siphon which includes no nonempty trap

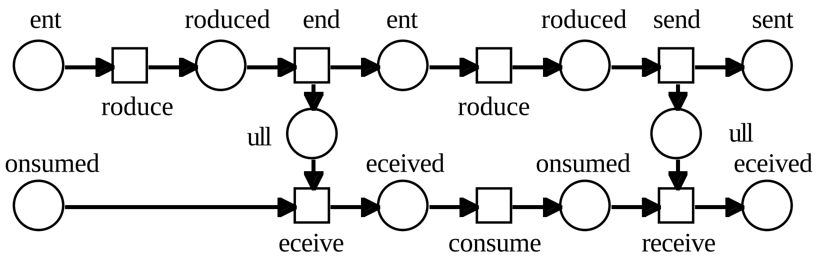
$\Rightarrow$ this free-choice net is not live.

Causal semantics of marked p/t-nets

Example: a producer / consumer system



a causal run of the producer /consumer system



Causal runs

A ***causal run*** of a marked p/t-net is given by a labeled Petri net (B, E, K)

Interpretation of causal runs

net element	name	symbol	interpretation
places	conditions	B	tokens on system places
transitions	events	E	system transition occurrences
arcs	causal relation	K	flow of tokens

Occurrence nets

An ***occurrence net*** is a net (B, E, K) with the following properties

- it has no cycles (i.e. K^+ is a partial order $\prec$)
- it has no branched places, i.e.
  $|^\bullet b|, |b^\bullet| \leq 1$ for each condition b
- events have finite fan-in and fan-out, i.e.
  $^\bullet e$ and $e^\bullet$ are finite sets for each event e
- it has neither input nor output-events, i.e.
  $|^\bullet e|, |e^\bullet| \geq 1$ for each event e
- no node has infinitely many predecessors, i.e.
  the set $\{x \in (B \cup E) \mid x \prec y\}$ is finite for each node y

Process nets of marked p/t-nets represent causal runs

Assume a marked p/t-net (S, T, F, k, w, m_0) without capacity restrictions

An occurrence net (B, E, K) together with
labels $\pi\colon (B \cup E) \to (S \cup T)$ is a ***process net*** of N if

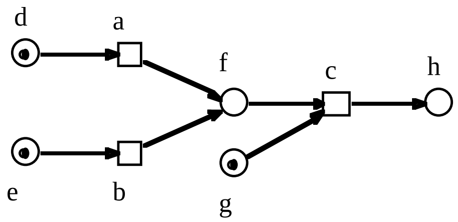
- sorts of nodes are respected by π, i.e.
  $\pi(B) \subseteq S$ and $\pi(E) \subseteq T$
- m_0 agrees with min(B), i.e.
  $m_0(s) = |\{b \in B \mid {}^\bullet b = \emptyset$ and $\pi(b) = s\}|$ for every place $s \in S$
- transition vicinities are preserved, i.e.
  $\pi(^\bullet e) = {}^\bullet(\pi(e))$, $|\{b \in {}^\bullet e \mid \pi(b) = s\}| = w(s, \pi(e))$ for each event e
  $\pi(e^\bullet) = (\pi(e))^\bullet$, $|\{b \in e^\bullet \mid \pi(b) = s\}| = w(\pi(e), s)$ for each event e

Occurrence sequences versus process nets

occurrence sequences provide total orders of events that respect causality but add arbitrary interleavings of independent events. Information about causal relationships can get lost.

process nets provide partial orders reflecting causality.

Example:



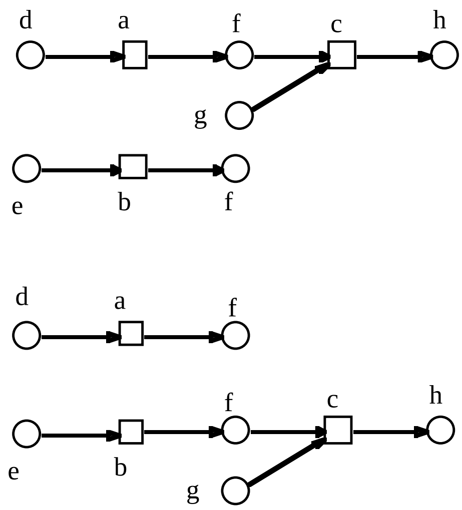
$a\ b\ c$

$b\ a\ c$

$a\ c\ b$

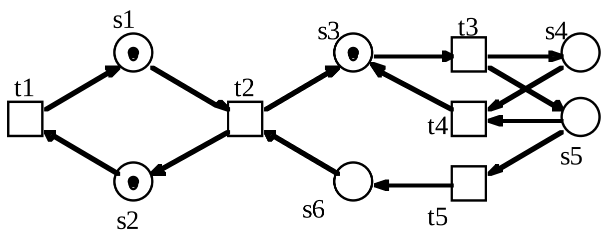
$b\ c\ a$

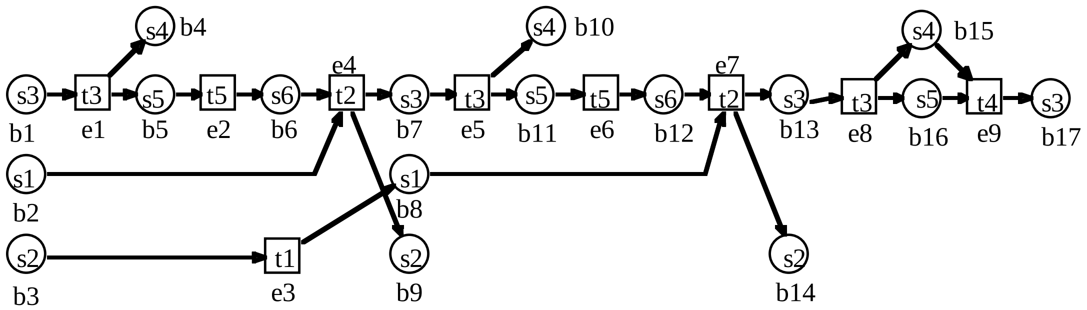
maximal occurrence sequences

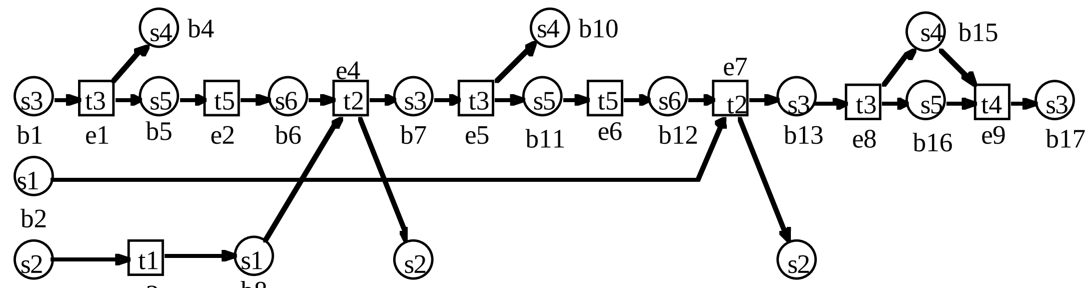


maximal process nets

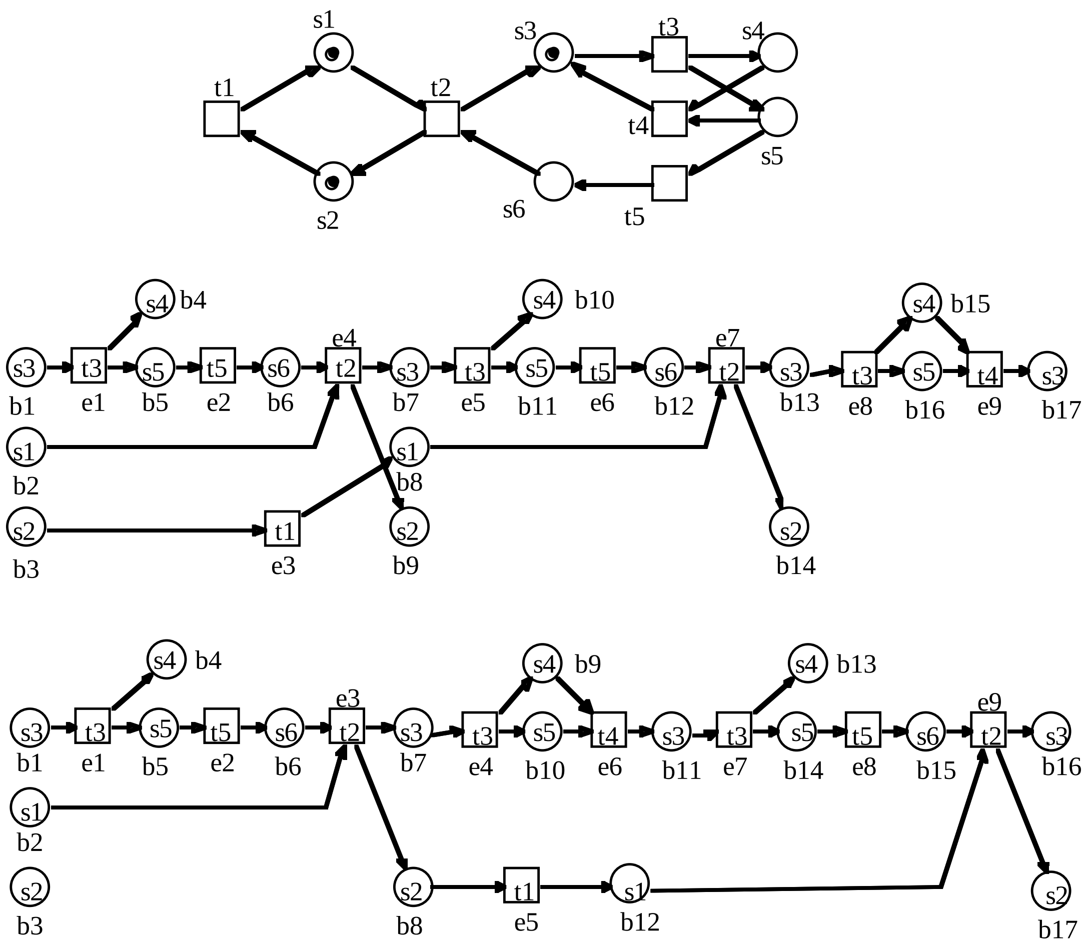
Two process nets corresponding to $t_1\ t_3\ t_5\ t_2\ t_3\ t_5\ t_2\ t_3\ t_4$







Two process nets without a common occurrence sequence



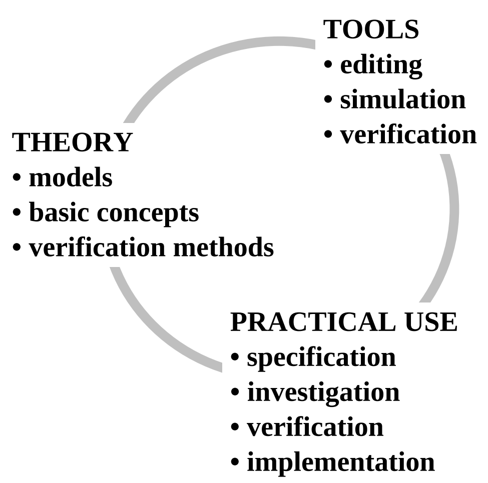
Coloured Petri Nets

Kurt Jensen

Computer Science Department
University of Aarhus

Ny Munkegade, Building 540
DK-8000 Aarhus C, Denmark

Phone: +45 89 42 32 34
Telefax: +45 89 42 32 55
E-mail: kjensen@daimi.au.dk
URL: http://www.daimi.au.dk/~kjensen



Part 1: Introduction to CP-nets

An ordinary Petri net (PT-net) has *no types* and *no modules:*

- Only one kind of tokens and the net is flat.

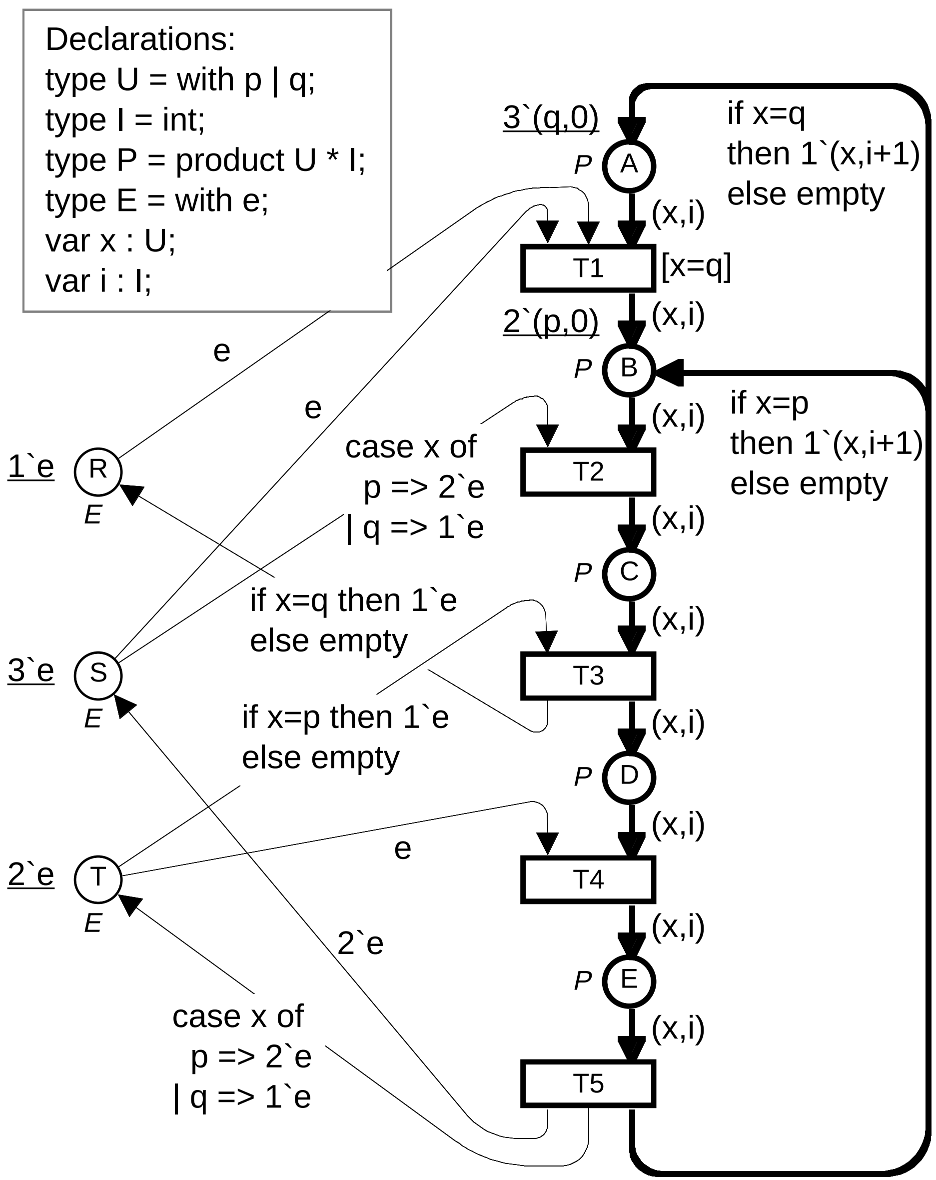
With Coloured Petri Nets (CP-nets) it is possible to use *data types* and complex *data manipulation:*

- Each token has attached a data value called the *token colour*.
- The token colours can be *investigated* and *modified* by the occurring transitions.

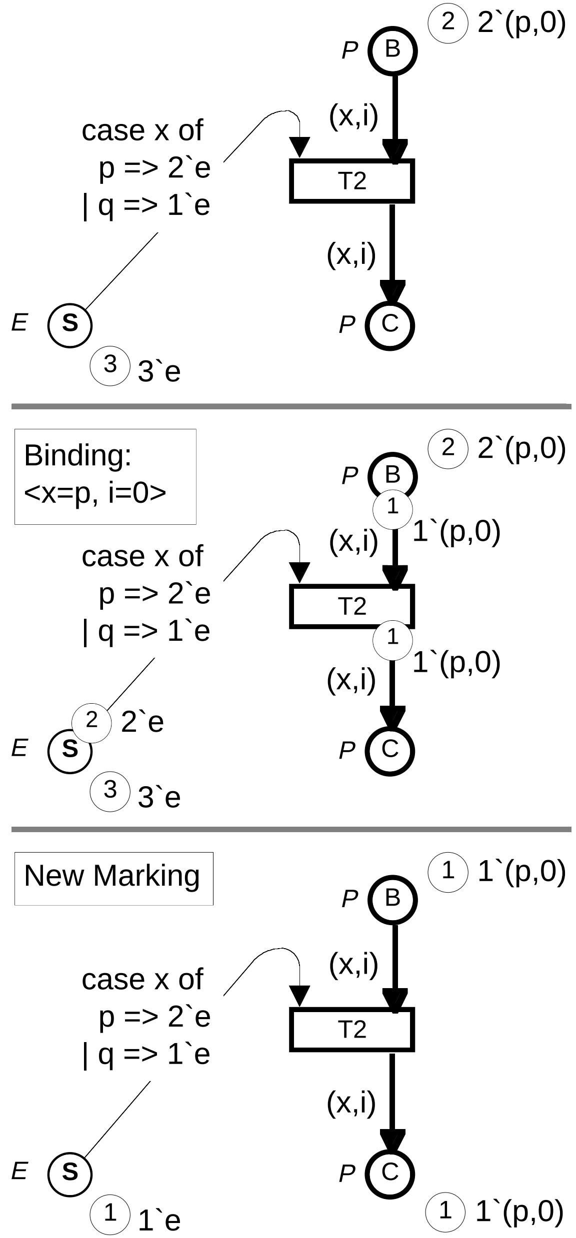
With CP-nets it is possible to make *hierarchical* descriptions:

- A large model can be obtained by *combining* a set of *submodels*.
- Well-defined *interfaces* between the submodels.
- Well-defined *semantics* of the combined model.
- *Submodels* can be reused.

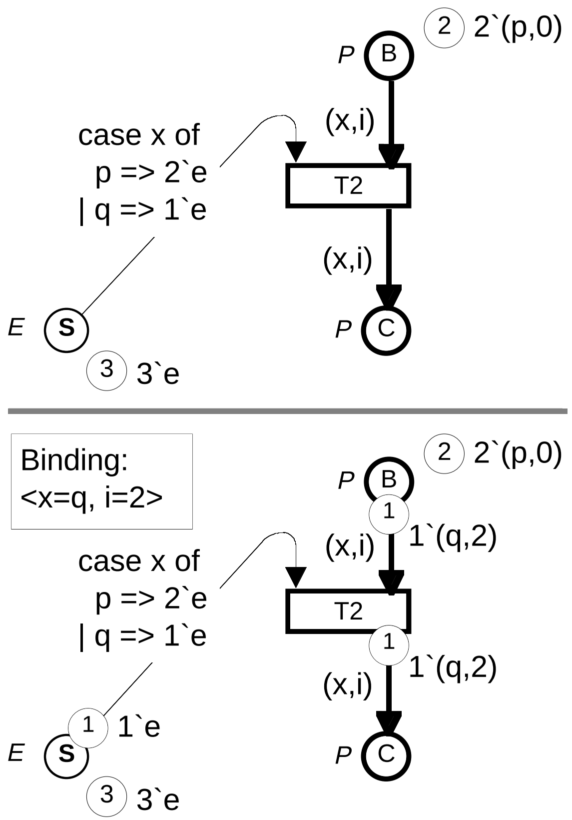
Resource allocation example



Occurrence of enabled binding

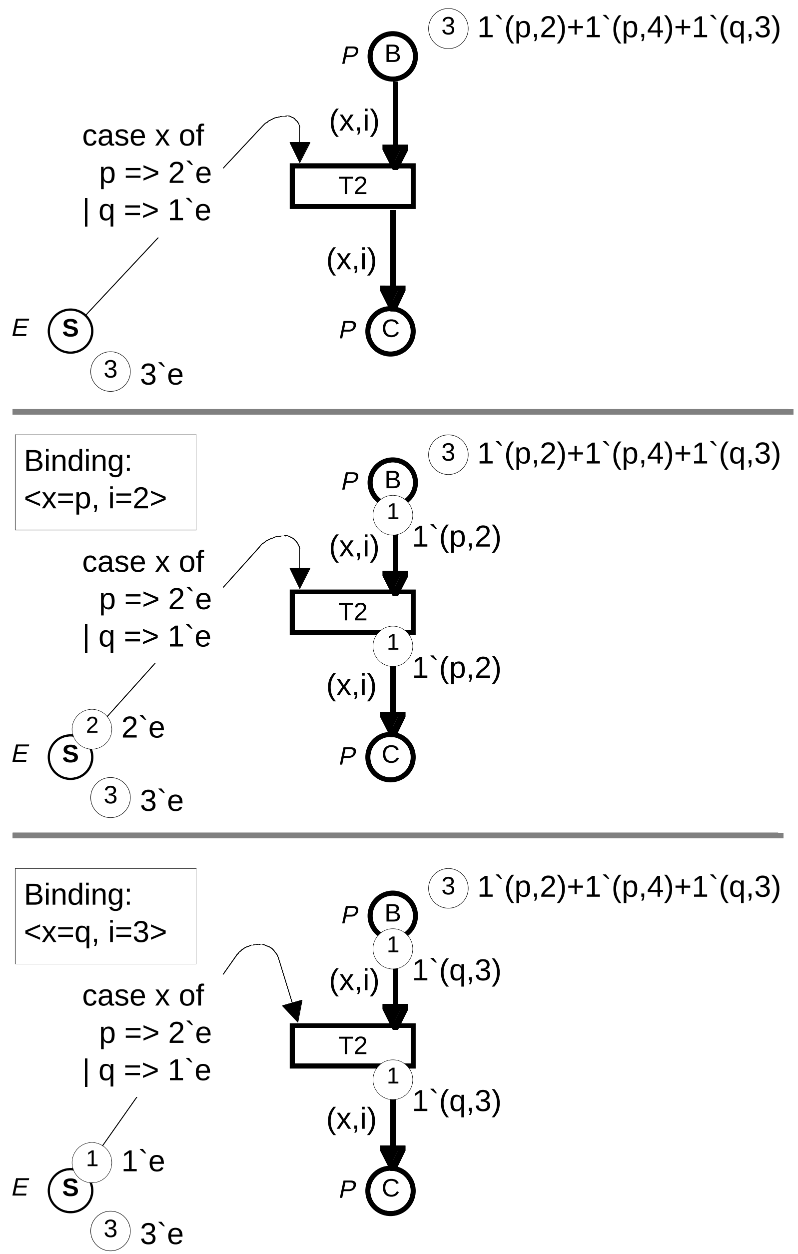


Binding which is not enabled

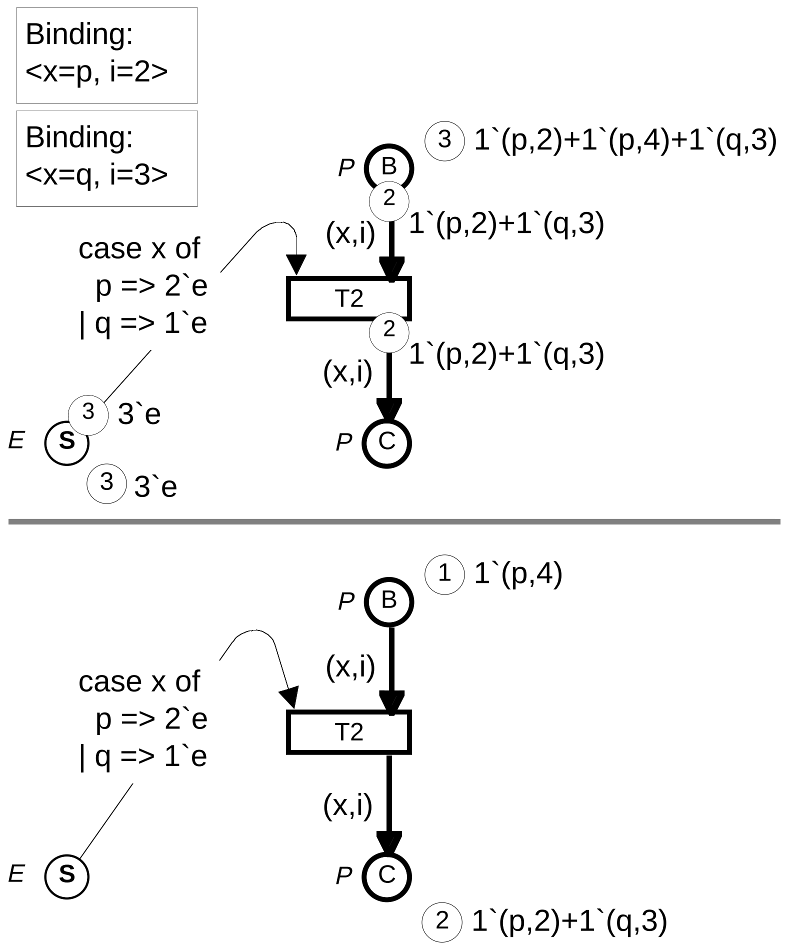


Binding cannot occur

A more complex example

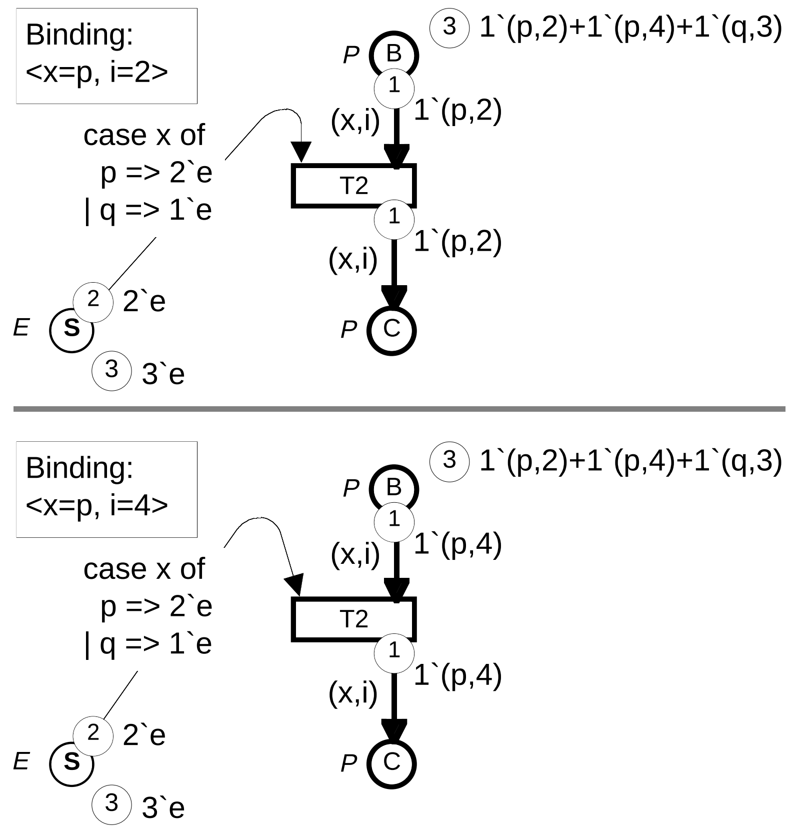


Concurrency



- The two bindings may occur *concurrently*.
- This is possible because they use *different tokens*.

Conflict



- These two bindings cannot occur *concurrently*.
- The reason is that they need the *same tokens*.

Resource allocation system

Two kinds of *processes:*

- Three cyclic q-processes (states A,B,C,D and E).
- Two cyclic p-processes (states B,C,D and E).

Three kinds of *resources:*

- Represented by the places R, S and T.

During a *cycle* a process *reserves* some resources and *releases* them again:

- Tokens are *removed* from and *added* to the resource places R, S and T.

A *cycle counter* is increased each time a process completes a full cycle.

It is rather straightforward to prove that the resource allocation system *cannot deadlock.*

- What happens if we add an additional token to place S – i.e., if we start with four S-resources instead of three?

Coloured Petri Nets

Declarations:

- *Types, functions, operations* and *variables.*

Each *place* has the following inscriptions:

- *Name* (for identification).
- *Colour set* (specifying the type of tokens which may reside on the place).
- *Initial marking* (multi-set of token colours).

Each *transition* has the following inscriptions:

- *Name* (for identification).
- *Guard* (boolean expression containing some of the variables).

Each *arc* has the following inscriptions:

- *Arc expression* (containing some of the variables). When the arc expression is evaluated it yields a multi-set of token colours.

Enabling and occurrence

A *binding* assigns a *colour* (i.e., a value) to each *variable* of a transition.

A *binding element* is a pair (t,b) where t is a *transition* while b is a *binding* for the variables of t. Example: (T2,<x=p, i=2>).

A binding element is *enabled* if and only if:

- There are *enough tokens* (of the correct colours on each input-place).
- The *guard* evaluates to true.

When a binding element is enabled it may *occur:*

- A multi-set of tokens is *removed* from each input-place.
- A multi-set of tokens is *added* to each output-place.

A binding element may occur *concurrently* to other binding elements – iff there are so many tokens that each binding element can get its "own share".

Main characteristics of CP-nets

Combination of *text* and *graphics.*

Declarations and *net inscriptions* are specified by means of a formal language, e.g., a *programming language.*

- Types, functions, operations, variables and expressions.

Net structure consists of places, transitions and arcs (forming a bi-partite graph).

- To make a CP-net *readable* it is important to make a nice graphical layout.
- The graphical layout has *no formal meaning.*

CP-nets have the same kind of *concurrency properties* as Place/Transition Nets.

Formal definition of CP-nets

Definition: A Coloured Petri Net is a tuple CPN = (Σ, P, T, A, N, C, G, E, I) satisfying the following requirements:

(i) Σ is a finite set of non-empty types, called **colour sets**.

(ii) P is a finite set of **places**.

(iii) T is a finite set of **transitions**.

(iv) A is a finite set of **arcs** such that:

- $P \cap T = P \cap A = T \cap A = \emptyset$.

(v) N is a **node** function. It is defined from A into $P \times T \cup T \times P$.

(vi) C is a **colour** function. It is defined from P into Σ.

(vii) G is a **guard** function. It is defined from T into expressions such that:

- $\forall t \in T$: [Type(G(t)) = Bool $\wedge$ Type(Var(G(t))) $\subseteq$ Σ].

(viii) E is an **arc expression** function. It is defined from A into expressions such that:

- $\forall a \in A$: [Type(E(a)) = $C(p(a))_{MS}$ $\wedge$ Type(Var(E(a))) $\subseteq$ Σ] where p(a) is the place of N(a).

(ix) I is an **initialization** function. It is defined from P into closed expressions such that:

- $\forall p \in P$: [Type(I(p)) = $C(p)_{MS}$].

Formal definition of behaviour

Definition: A **step** is a multi-set of binding elements.

A step Y is **enabled** in a marking M iff the following property is satisfied:

$$\forall p \in P: \sum_{(t,b) \in Y} E(p,t)\langle b\rangle \leq M(p).$$

When a step Y is enabled in a marking M_1 it may **occur**, changing the marking M_1 to another marking M_2, defined by:

$$\forall p \in P: M_2(p) = (M_1(p) - \sum_{(t,b) \in Y} E(p,t)\langle b\rangle) + \sum_{(t,b) \in Y} E(t,p)\langle b\rangle.$$

The first sum is called the **removed** tokens while the second is called the **added** tokens. Moreover we say that M_2 is **directly reachable** from M_1 by the occurrence of the step Y, which we also denote: $M_1[Y\rangle M_2$.

An **occurrence sequence** is a sequence of markings and steps:

$$M_1[Y_1\rangle M_2[Y_2\rangle M_3 \ldots M_n[Y_n\rangle M_{n+1}$$

such that $M_i[Y_i\rangle M_{i+1}$ for all $i \in 1..n$. We then say that M_{n+1} is **reachable** from M_1. We use $[M\rangle$ to denote the set of markings which are reachable from M.

Formal definition

The existence of a *formal definition* is very important:

- It is the basis for *simulation*, i.e., execution of the CP-net.
- It is also the basis for the *formal verification* methods (e.g., state spaces and place invariants).
- Without the formal definition, it would have been impossible to obtain a *sound* net class.

It is *not necessary* for a *user* to know the formal definition of CP-nets:

- The correct *syntax* is checked by the CPN editor, i.e., the computer tool by which CP-nets are constructed.
- The correct use of the *semantics* (i.e., the enabling rule and the occurrence rule) is guaranteed by the CPN simulator and the CPN tools for formal verification.

High-level contra low-level nets

The relationship between CP-nets and Place/Transition Nets (PT-nets) is *analogous* to the relationship between high-level programming languages and assembly code.

- In theory, the two levels have exactly the same *computational power*.
- In practice, high-level languages have much more *modelling power* – because they have better structuring facilities, e.g., types and modules.

Each CP-net has an *equivalent* PT-net – and vice versa.

- This equivalence is used to derive the definition of *basic properties* and to establish the *verification methods*.
- In practice, we *never* translate a CP-net into a PT-net – or vice versa.
- Description, simulation and verification are done *directly* in terms of CP-nets.

Other kinds of high-level nets

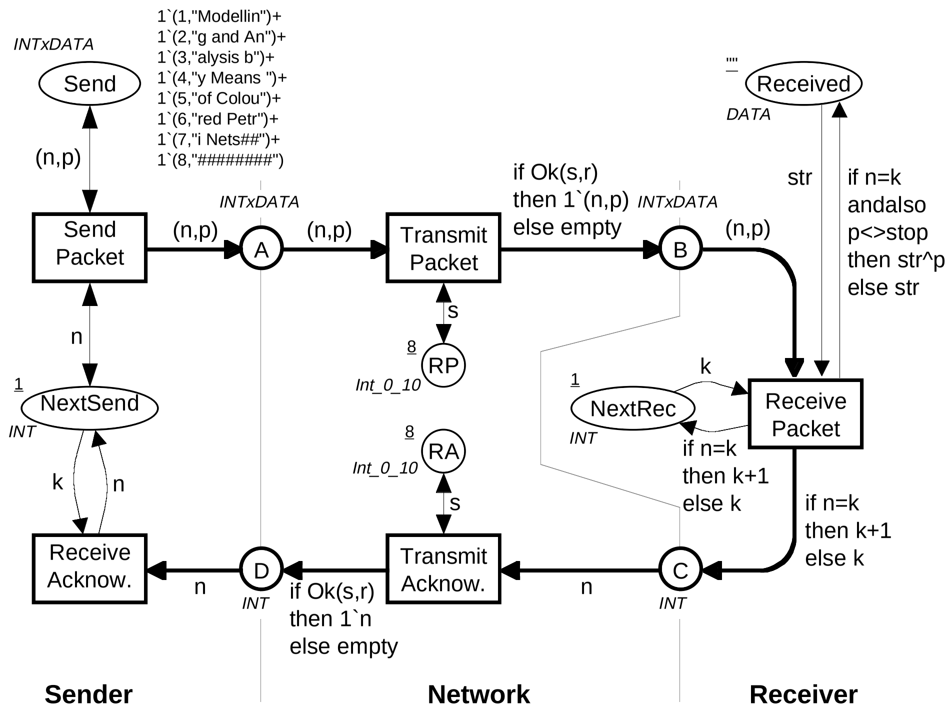
CP-nets have been developed from Predicate/Transition Nets.

- *Hartmann Genrich & Kurt Lautenbach*.
- With respect to *description* and *simulation* the two models are nearly identical.
- With respect to *formal verification* there are some differences.

Several other kinds of high-level Petri Nets exist.

- They all build upon the same *basic ideas*, but use *different languages* for declarations and inscriptions.
- A detailed comparison is outside the scope of this talk.

Simple protocol

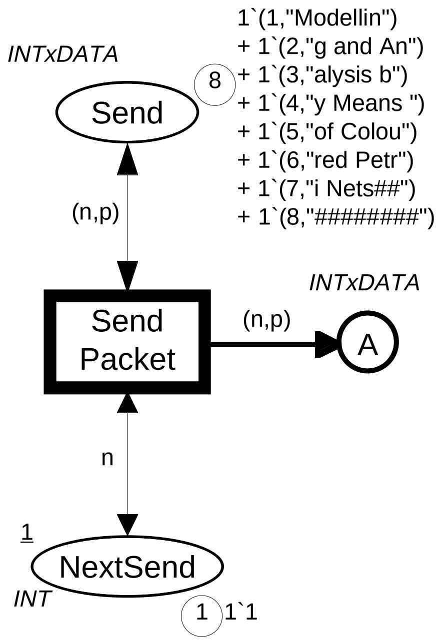



```
type INT = int;
type BOOL = bool;
type DATA = string;
type INTxDATA = product INT * DATA;
var n, k : INT;
var p,str : DATA;
val stop = "########";

type Int_0_10 = int with 0..10;
type Int_1_10 = int with 1..10;
var s : Int_0_10;
var r : Int_1_10;

fun Ok(s : Int_0_10, r : Int_1_10) = (r≤s);
```

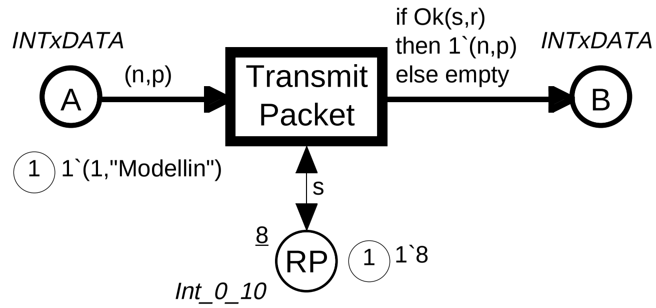
Send packet



Only the binding <n=1, p="Modellin"> is *enabled.*

- When the binding *occurs* it adds a token to place A. The token represents that the packet (1,"Modellin") is sent to the network.
- The packet is *not* removed from place *Send* and the *NextSend* counter is *not* changed.

Transmit packet



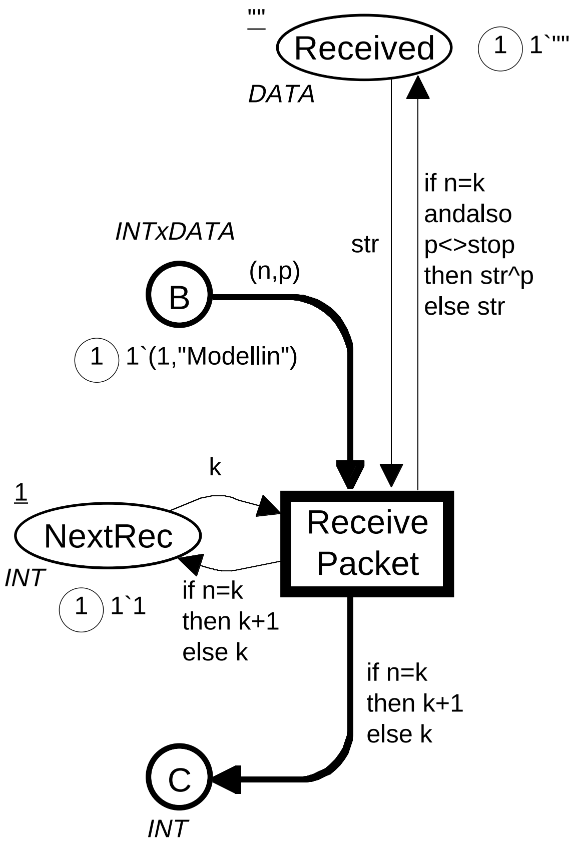
There are now *10 enabled bindings:*

- They are all of the form
  <n=1, p="Modellin", s=8, r=…>.
- The variable r can take *10 different values*, because the type of r is defined to contain the integers 1..10.

The *function* Ok(s,r) checks whether r ≤ s.

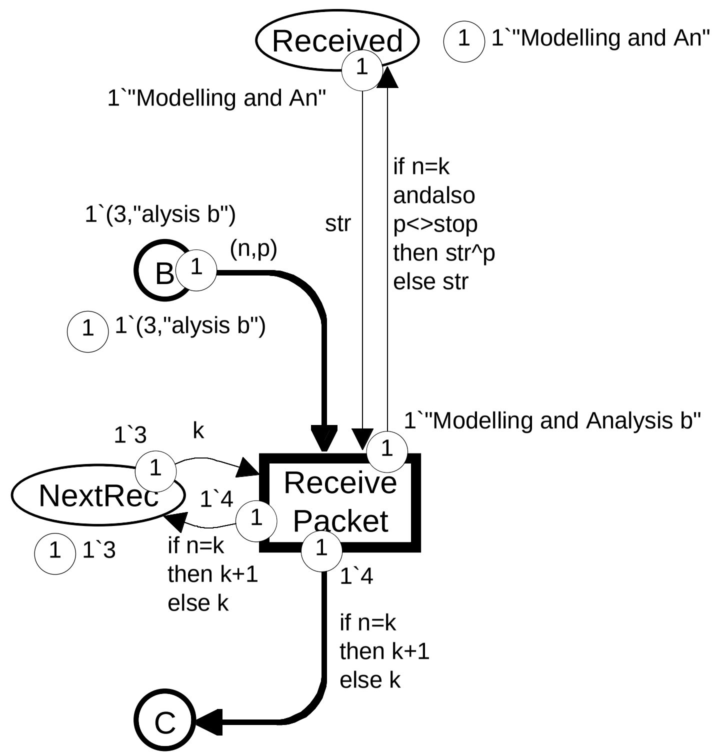
- For r∈1..8, *Ok(s,r) = true*. This means that the token is moved from A to B, i.e., that the packet is *successfully transmitted* over the network.
- For r∈9..10, *Ok(s,r) = false*. This means that no token is added to B, i.e., that the packet is *lost*.
- The CPN simulator make *random* choices between enabled bindings. Hence there is *80%* chance for *successful transfer*.

Receive packet



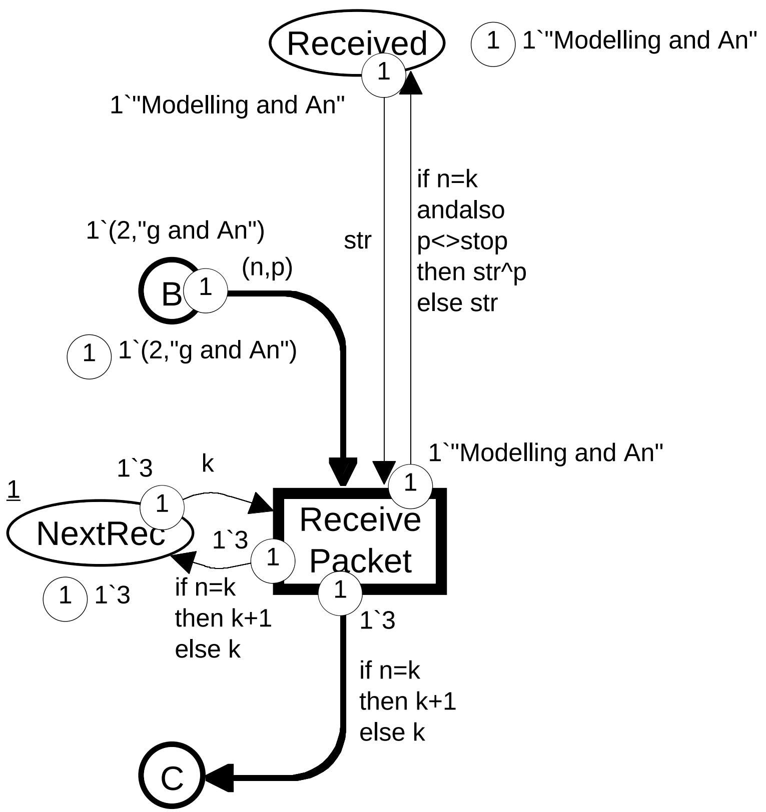
It is checked whether the number of the incoming packet n *matches* the number of the expected packet k.

Correct packet number



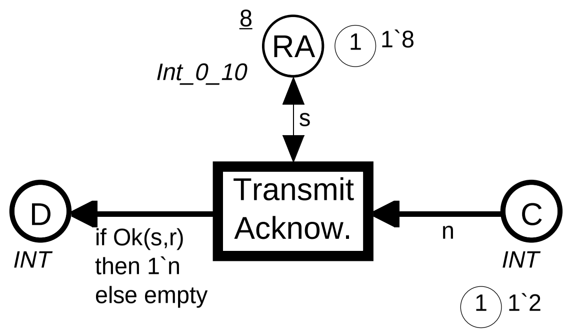
- The data in the packet is *concatenated* to the data already received.
- The *NextRec* counter is increased by one.
- An *acknowledgement message* is sent. It contains the number of the next packet which the receiver wants to get.

Wrong packet number



- The data in the packet is *ignored*.
- The *NextRec* counter is unchanged.
- An *acknowledgement message* is sent. It contains the number of the next packet which the receiver wants to get.

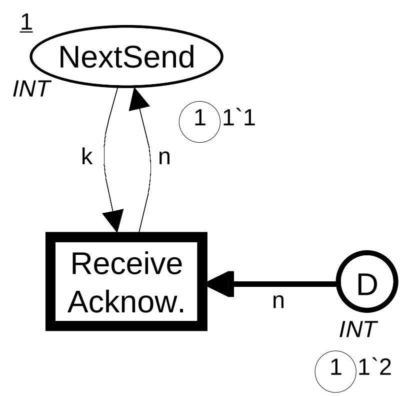
Transmit acknowledgement



This transition works in a similar way as *Transmit Packet*.

- The token on place *RA* determines the success rate for transmission of acknowledgements.
- When *RA* contains a token with value 8, the success rate is *80%*.
- When *RA* contains a token with value 10, *no* acknowledgements are lost.
- When *RA* contains a token with value 0, *all* acknowledgements are lost.

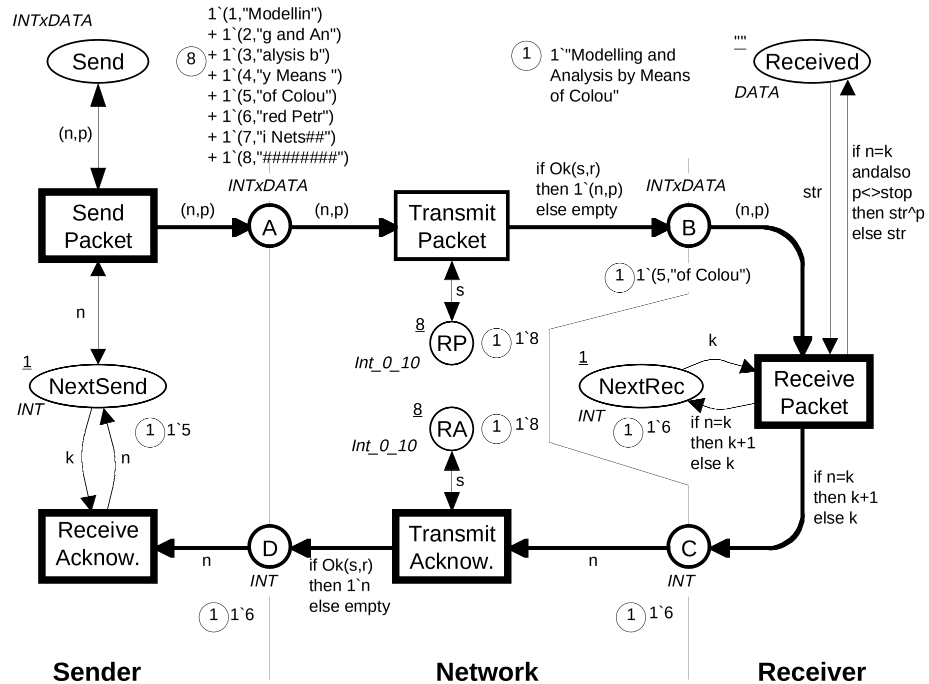
Receive acknowledgement



When an acknowledgement *arrives* to the *Sender* it is used to update the *NextSend* counter.

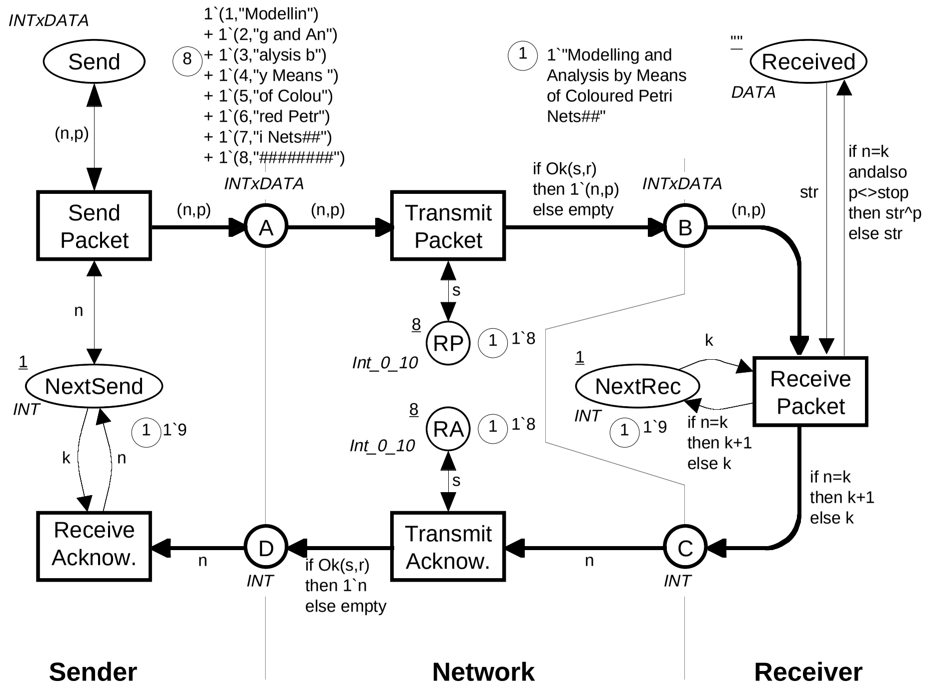
- In this case the counter value becomes 2, and hence the *Sender* will begin to send packet number 2.

Intermediate state



- The *Receiver* is expecting package no. 6. This means that it has successfully received the first 5 packets.
- The *Sender* is still sending packet no. 5. In a moment it will receive an acknowledgement containing a request for packet no. 6.
- When the acknowledgement is received the *NextSend* counter is updated and the *Sender* will start sending packet no. 6.

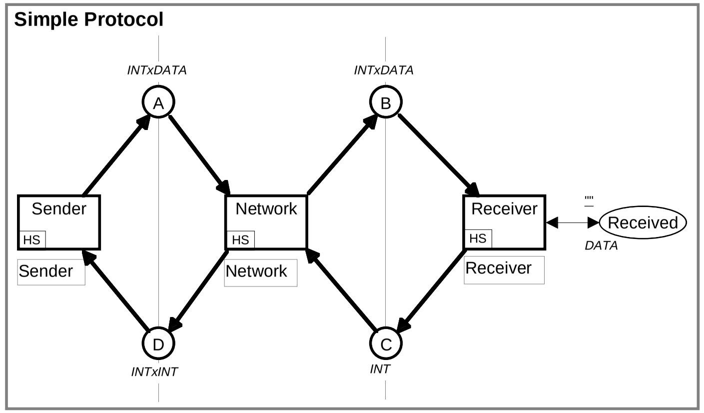
Final state

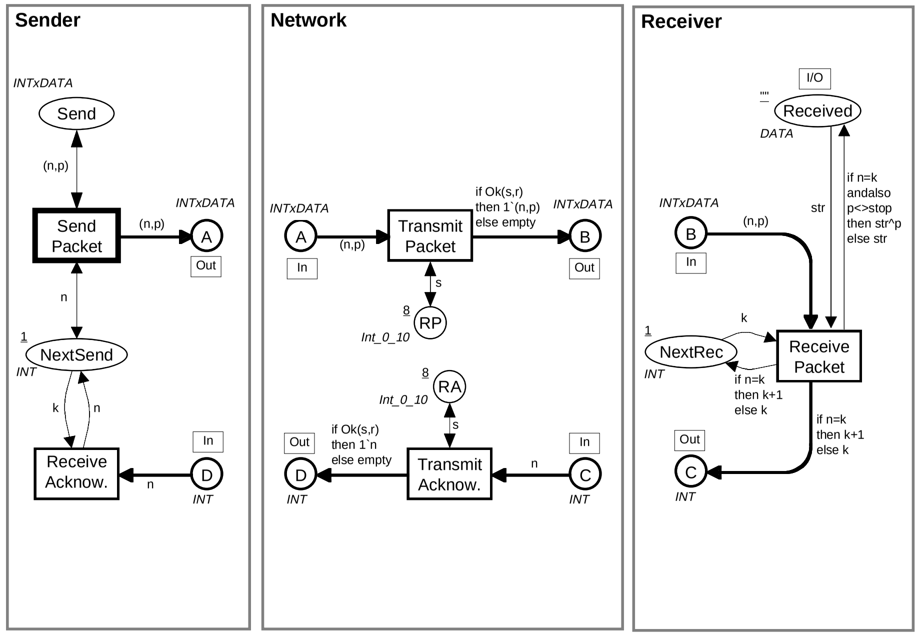


- When the last packet, i.e., packet no. 8 reaches the *Receiver* an acknowledgement with value 9 is sent.
- When this acknowledgement reaches the *Sender* the *NextSend* counter is updated to 9.
- This means that the *Send Packet* transition no longer can occur, and hence the transmission stops.

Part 2: Hierarchical CP-nets

A hierarchical CP-net contains a number of *interrelated subnets*– called *pages*.





Substitution transitions

A page may contain one ore more *substitution transitions*.

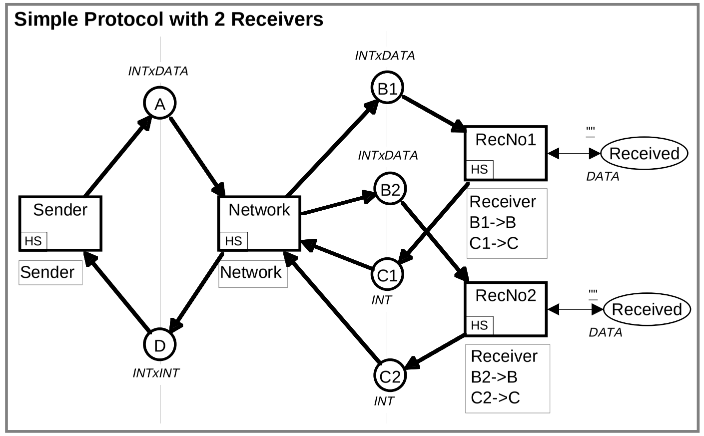
- Each substitution transition is related to a *page*, i.e., a *subnet* providing a *more detailed description* than the transition itself.
- The page is a *subpage* of the substitution transition.

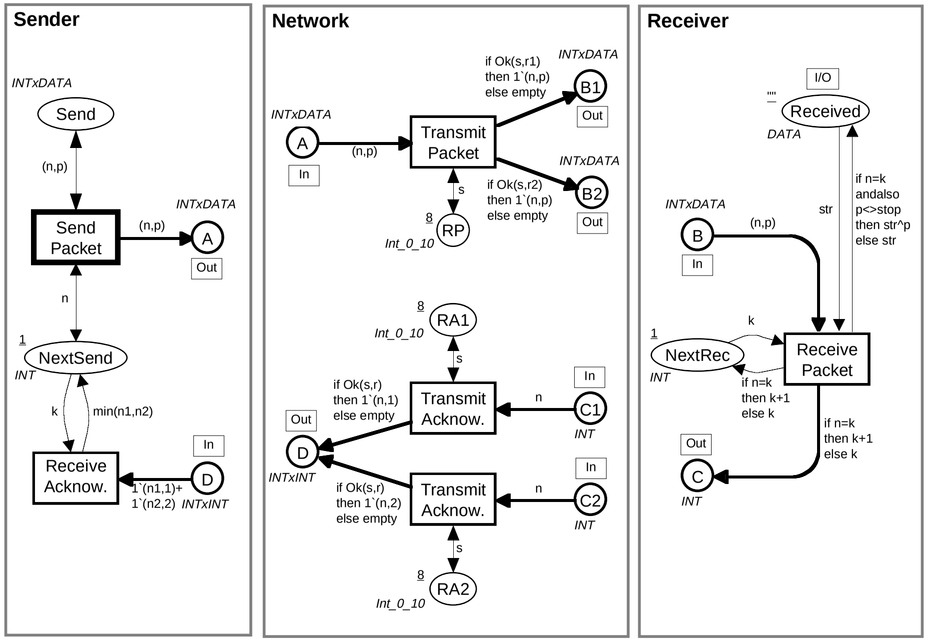
There is a *well-defined interface* between a substitution transition and its subpage:

- The places surrounding the substitution transition are *socket places*.
- The subpage contains a number of *port places*.
- Socket places are *related* to port places – in a similar way as actual parameters are related to formal parameters in a procedure call.
- A socket place has always the *same marking* as the related port place. The two places are just *different views* of the same place.

Substitution transitions work in a similar way as the refinement primitives found in many system description languages – e.g., SADT diagrams.

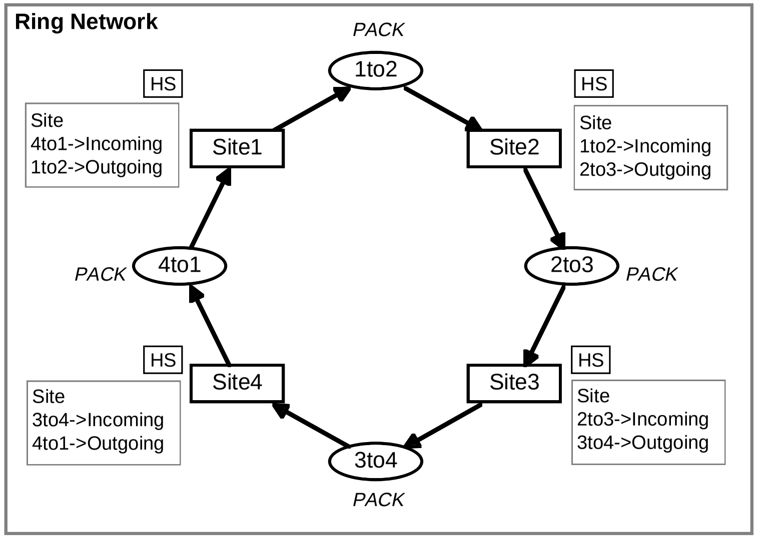
Pages can be used more than once

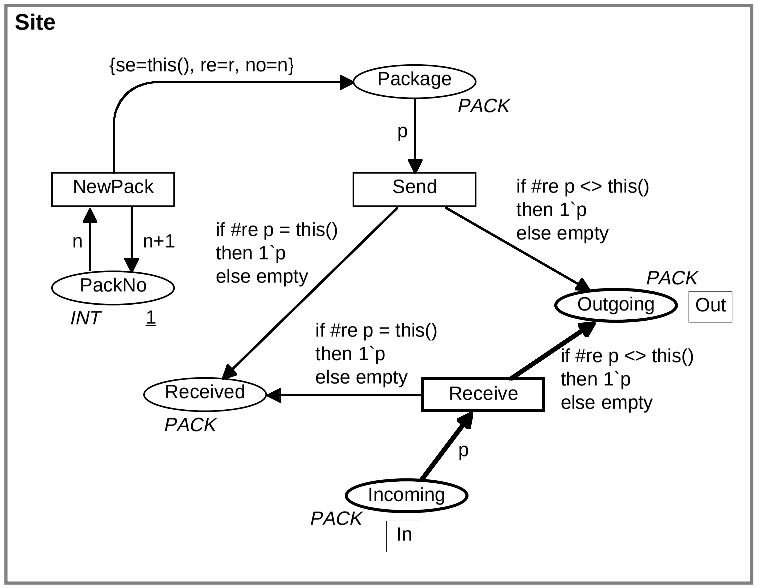




There are *two* different *instances* of the *Receiver* page. Each instance has its *own marking*.

Ring network





Formal definition of hierarchical CP-nets

The *syntax* and *semantics* of hierarchical CP-nets have *formal definitions* – similar to the definitions for non-hierarchical CP-nets

Each hierarchical CP-net has an *equivalent* non-hierarchical CP-net – and vice versa.

- The two kinds of nets have the same *computational power* – but hierarchical CP-nets have much more *modelling power*.
- The *equivalence* is used for *theoretical purposes*.
- In practice, we *never* translate a hierarchical CP-net into a non-hierarchical CP-net – or vice versa.

CP-nets may be large

A typical *industrial application* of CP-nets contains:

- 10-200 *pages*.
- 50-1000 *places and transitions.*
- 10-200 *colour sets*.

This corresponds to *thousands/millions of nodes* in a Place/Transition Net.

Most of the industrial applications would be *totally impossible* without:

- Colours.
- Hierarchies.
- Computer tools.

Protocol for telephone network

Transport layer of a protocol for *digital telephone communication*.

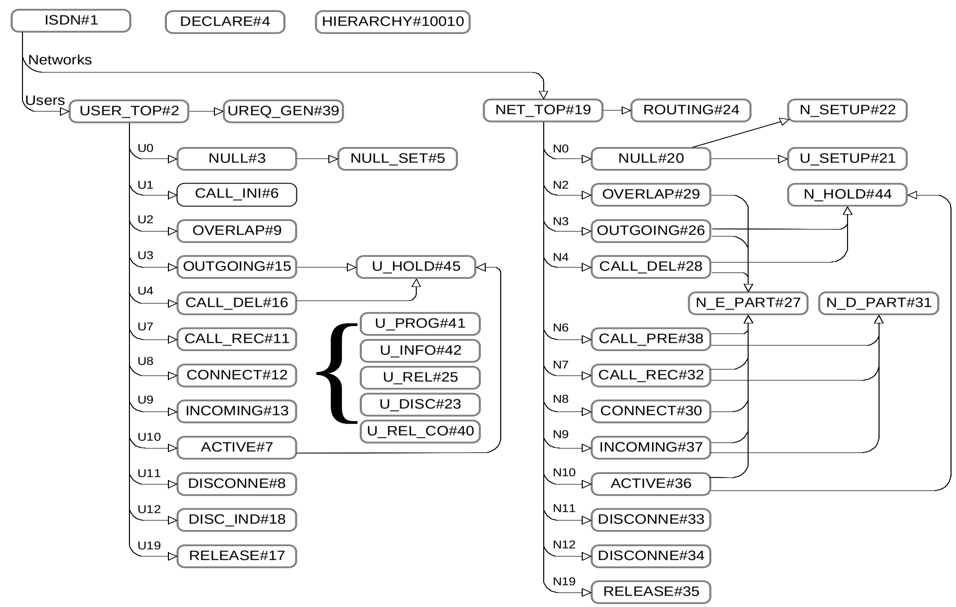
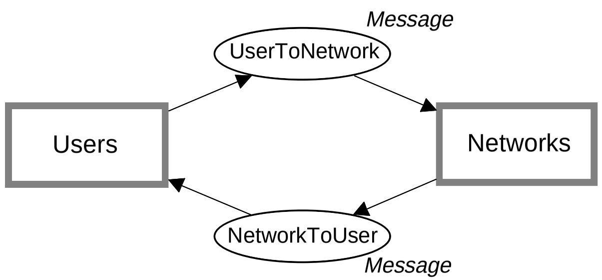


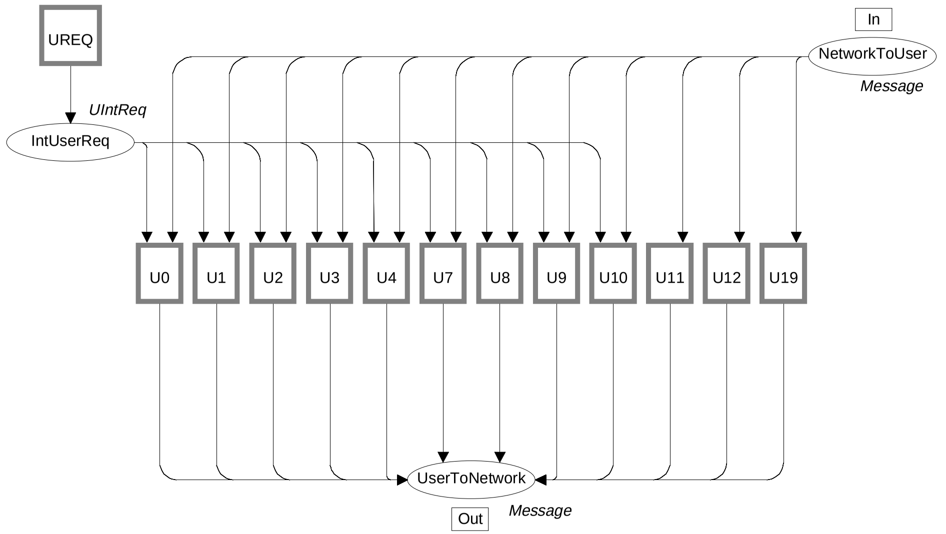
Fig. 7.1

Overview of the hierarchy structure:

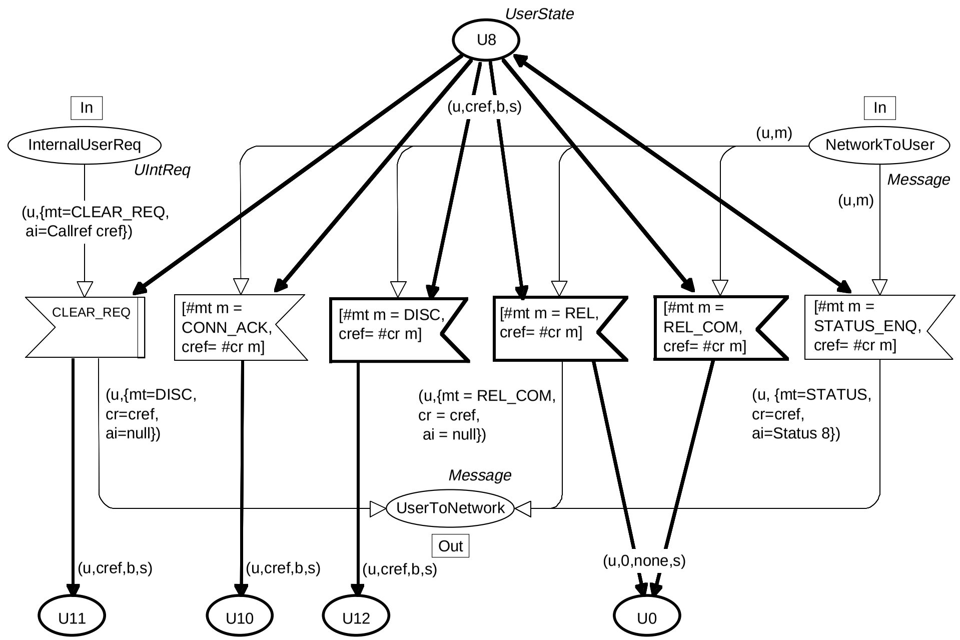
- Each *node* represents a *page*, i.e., a subnet.
- Each *arc* represents a *transition substitution.*

Two of the most abstract pages





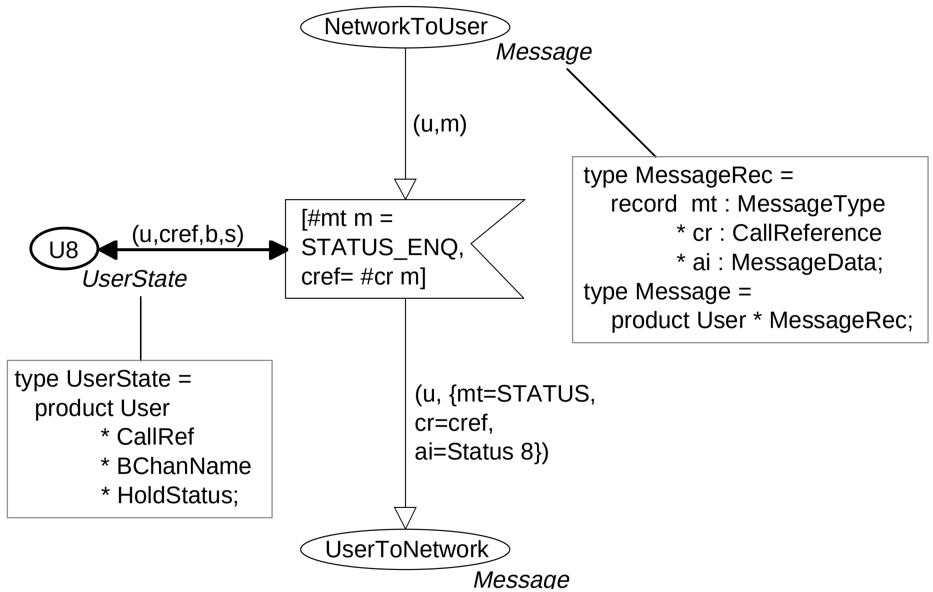
Typical page for the user site



This page describes the *possible actions* that can happen when the user site is in state *U8:*

- From the *network* five different kinds of messages may be received.
- In addition there is one kind of *internal user request.*
- In three of the cases a *new message* is sent to the *network site.*

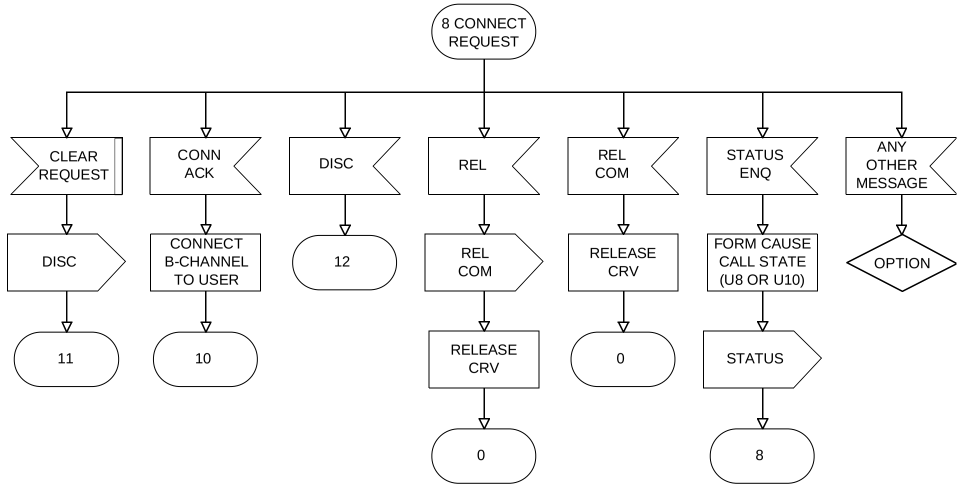
Typical transition



This transition describes the *actions* that are taken when a *Status Enquiry* message is received in state *U8:*

- The guard checks that the message is a *Status Enquiry* message. It also checks that the *Call Reference* is correct (i.e., matches the one in the *User State* token at place U8).
- A *Status message* is sent to the *network site.* It tells that the user site is in state U8.

SDL description of user page



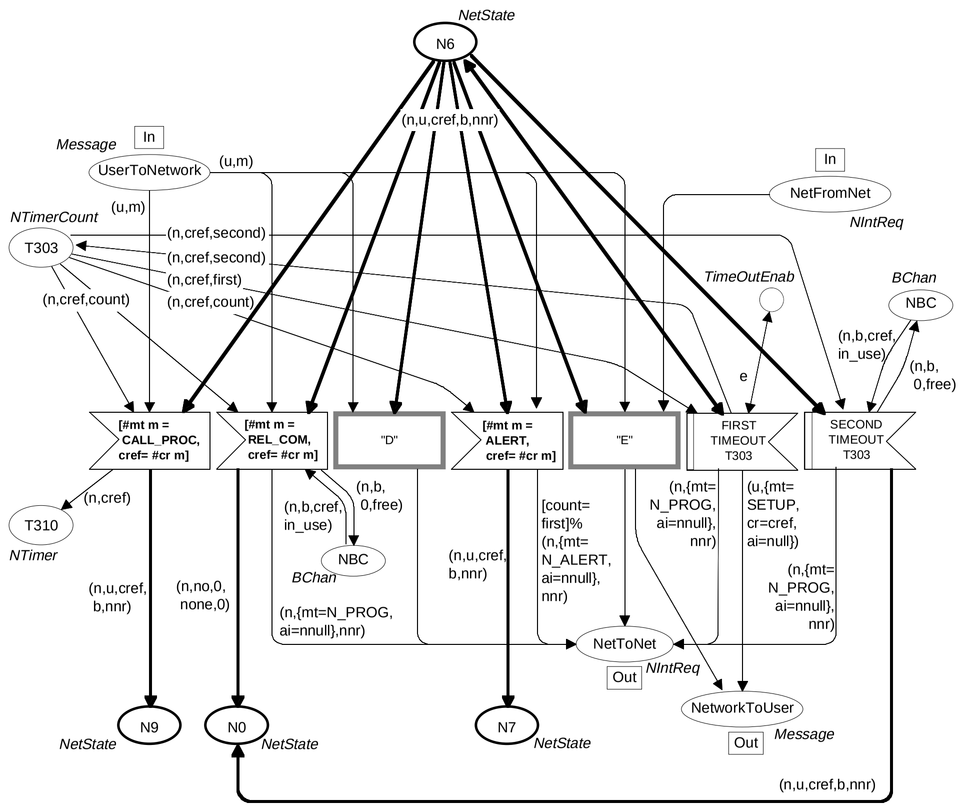
Each *vertical string of SDL symbols* describes a sequence of actions – which is translated into a *single CPN transition.*

- The *translation* from SDL to CPN was done *manually*.
- The translation is straightforward and it could easily be *automated.*

The graphical shape of a node has a *well-defined* meaning in SDL.

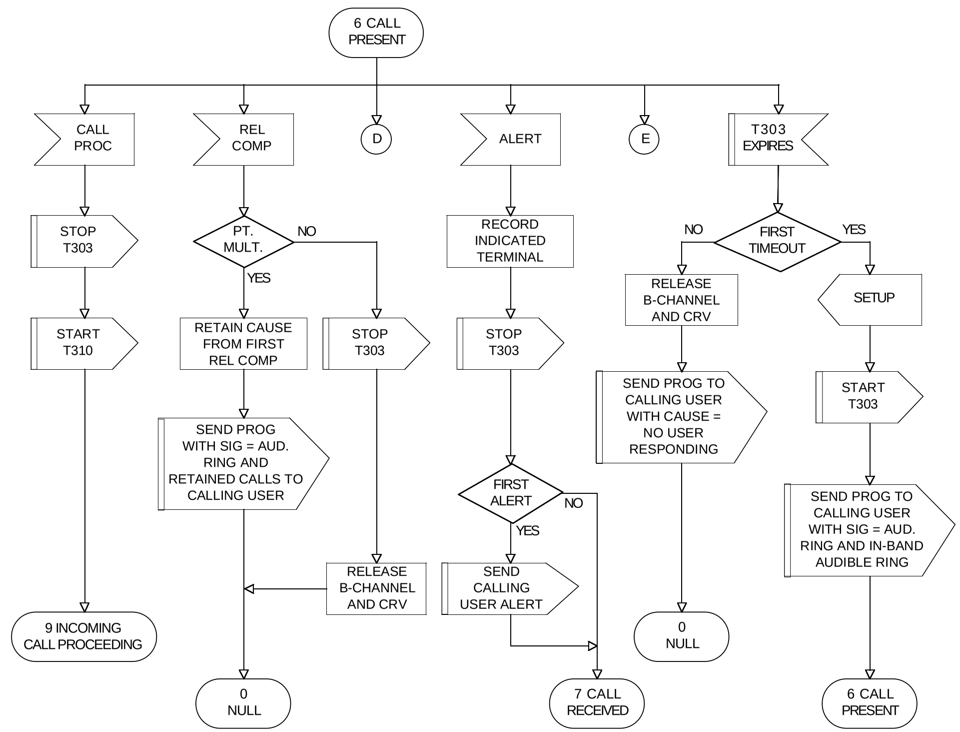
- In the CP-net the shape is retained – to improve the *readability*. It has no formal meaning.

Typical page for the network site



Similar structure as for the user page – but slightly more complex.

SDL description of network page



Similar structure as for the user page – but slightly more complex.

It is easy to see that there is a very straightforward relationship between the *SDL page* and the corresponding *CPN page*.

Some pages are used many times

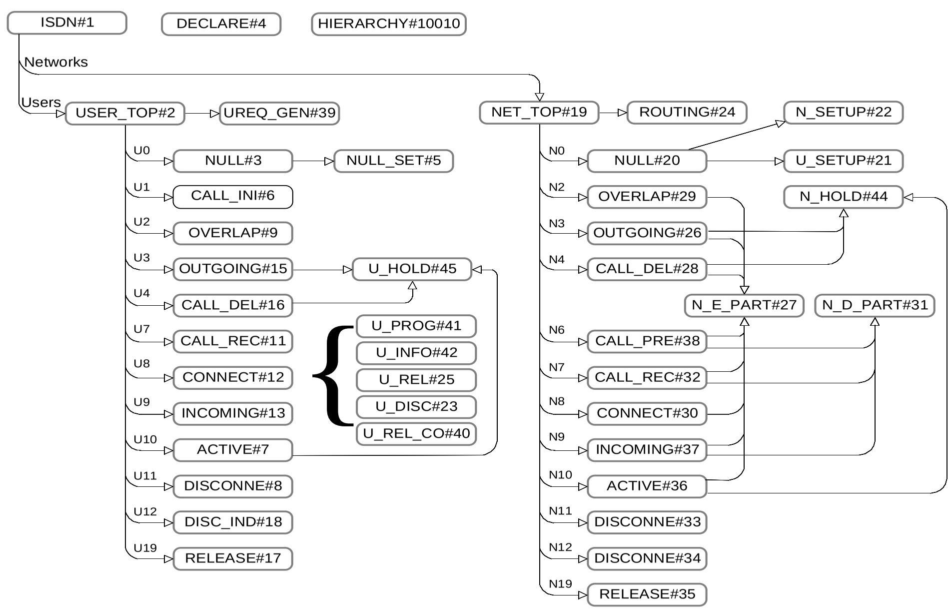


Fig. 7.1

- 43 pages with more than 100 page instances.
- The entire modelling of this – fairly complex protocol – was made in only *3 weeks* (by a single person).
- According to engineers at the participating telecommunications company, the CPN model was the *most detailed* behavioural model that they had ever seen for such protocols.

Practical use of CP-nets

CP-nets are used in *many different areas*. A few selected examples are:

- Communication protocols (BRI, DQDB, ATM).
- VLSI chips (clocked and self-timed).
- Banking procedures (check processing and funds transfer).
- Correctness of ADA programs (rendezvous structure).
- Teleshopping systems.
- Military systems (radar control post and naval vessel).
- Security systems (intrusion alarms, etc.).
- Flexible manufacturing.

Summary of practical experiences

Graphical representation and *executability* are extremely important.

Most practical models are *large*.

- They cannot be constructed without the *hierarchy concepts.*
- Neither can they be constructed or verified without the *computer tools.*

CP-nets are often used *together* with other graphical description languages, such as SADT, SDL and block diagrams.

- This means that the user does not have to learn a completely *new language.*

CP-nets are well-suited for *verification* of existing designs – in particular concurrent systems.

- CP-nets can also be used to *design* new systems.
- Then it is possible to use the *insight* gained through the modelling, simulation and verification activities – to *improve* the design itself.

Part 3: Construction and Simulation of CP-nets

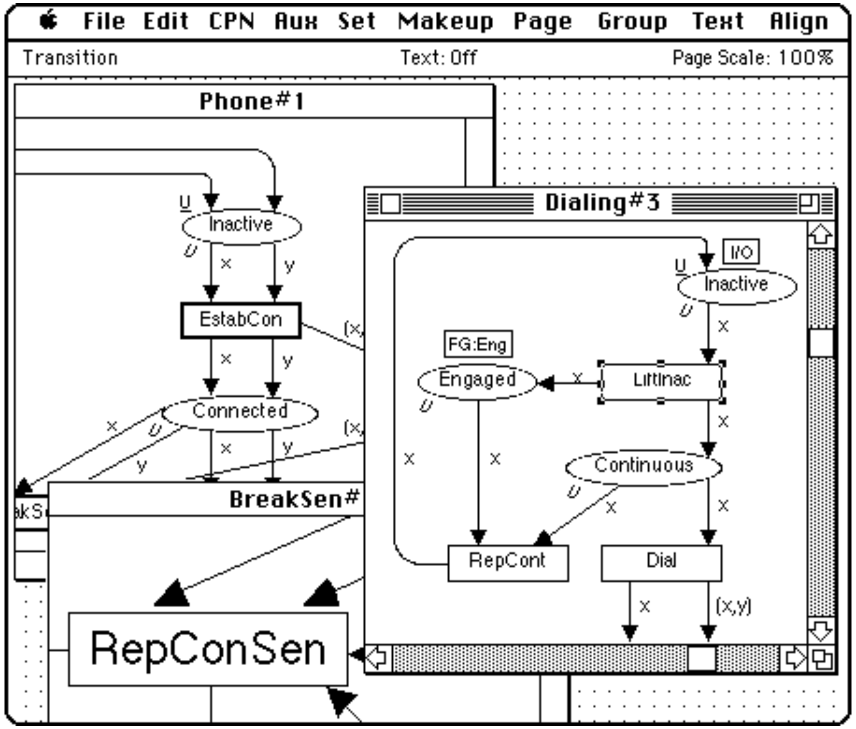
CP-nets have an *integrated* set of *robust* computer tools with *reliable support:*

- *Construction* and *modification* of CPN models.
- *Syntax checking* (e.g., types and module interfaces).
- *Interactive simulation*, e.g., to gain additional understanding of the modelled system. Can also be used for *debugging*.
- *Automatic simulations*, e.g., to obtain performance measures. Can also be used for *prototyping*.
- *Verification* to *prove* behavioural properties.
  - *State spaces* (also called reachability graphs and occurrence graphs).
  - *Place invariants*.

The computer tools are available on:

- Sun Sparc with Solaris.
- HP with HPUX
- Intel PCs with Linux.
- Macintosh with Mac OS.

CPN editor



Each *page* is shown in its own *window*.

The user performs an operation by selecting an *object* and a *command* for it, e.g.:

- Select a *transition* (by pointing with the mouse).
- Select the desired *command* (by pointing in the corresponding drop-down menu).

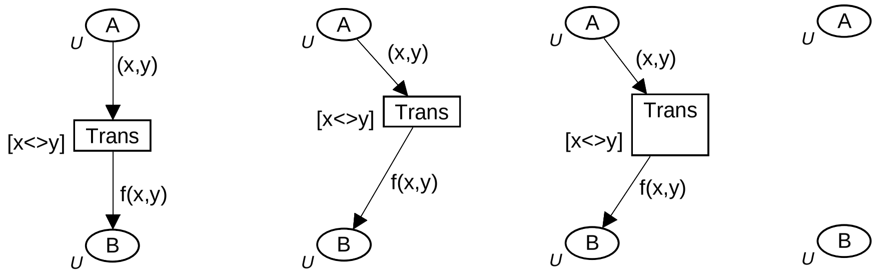
Commands can be performed on a *set of objects*.

Editor knows syntax of CP-nets

Some kinds of errors are *impossible*, e.g.:

- An arc between *two places* or *two transitions.*
- A place with *two colour sets* or an arc with *two arc expressions.*
- A transition with a *colour set.*
- Port assignment involving a place which is a *non-socket* or a *non-port.*
- A *cyclic* set of *substitution transitions.*

The editor behaves *intelligently:*



- When a node is *repositioned* or *resized* the surrounding arcs and inscriptions are *automatically adjusted.*
- When a node is *deleted* the surrounding arcs are *automatically deleted.*

Attributes

Each graphical object has its own *attributes*.

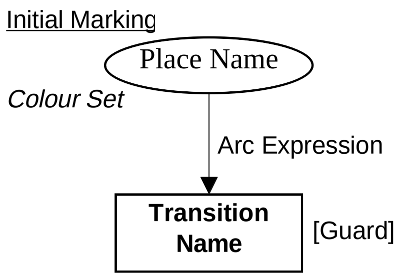
They determine how the object appears on the screen/print-outs:

- Text attributes

- Graphical attributes

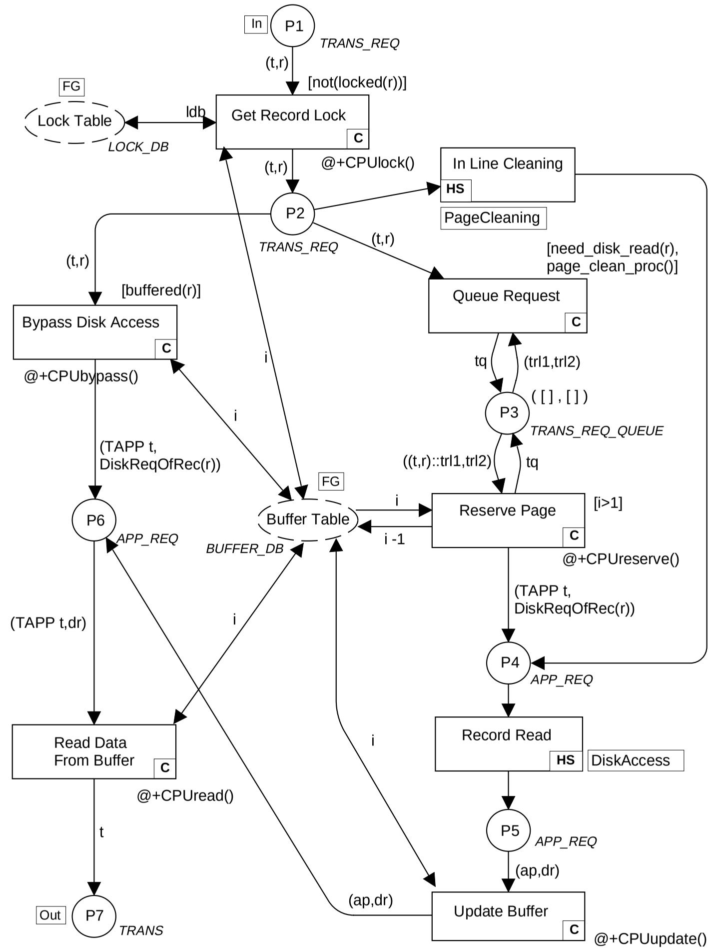
- Shape attributes

Each *kind of objects* has its own *defaults:*



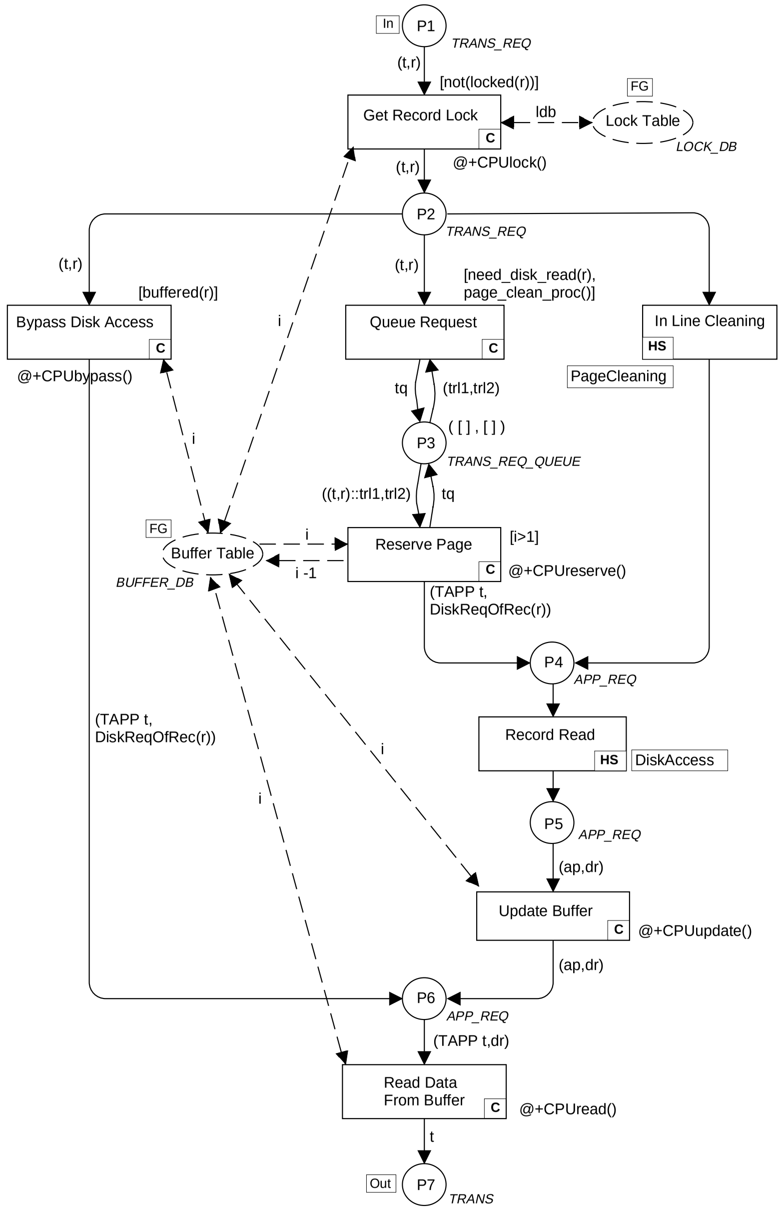
Defaults can be *changed* and they can be *overwritten* (for individual objects).

Easy to experiment

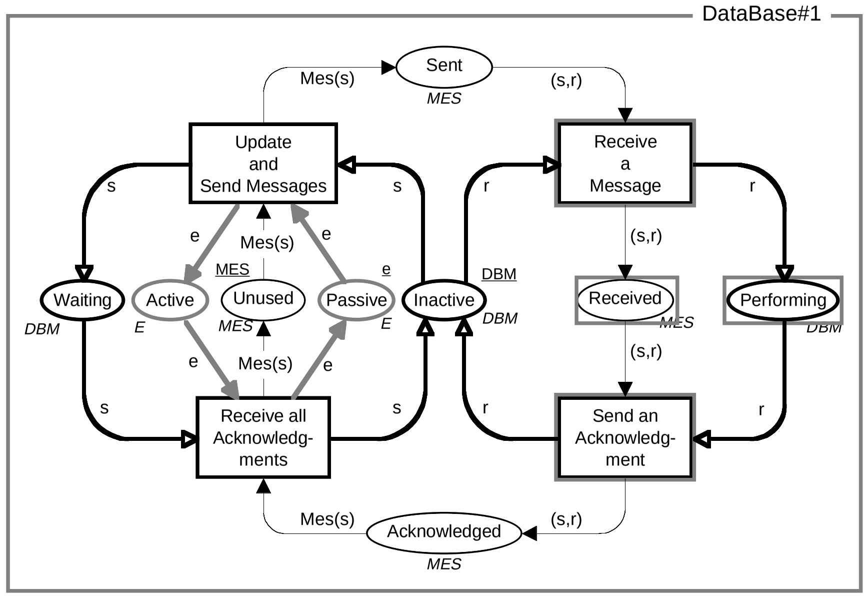


Can we improve the *layout* of this page?

Improved layout



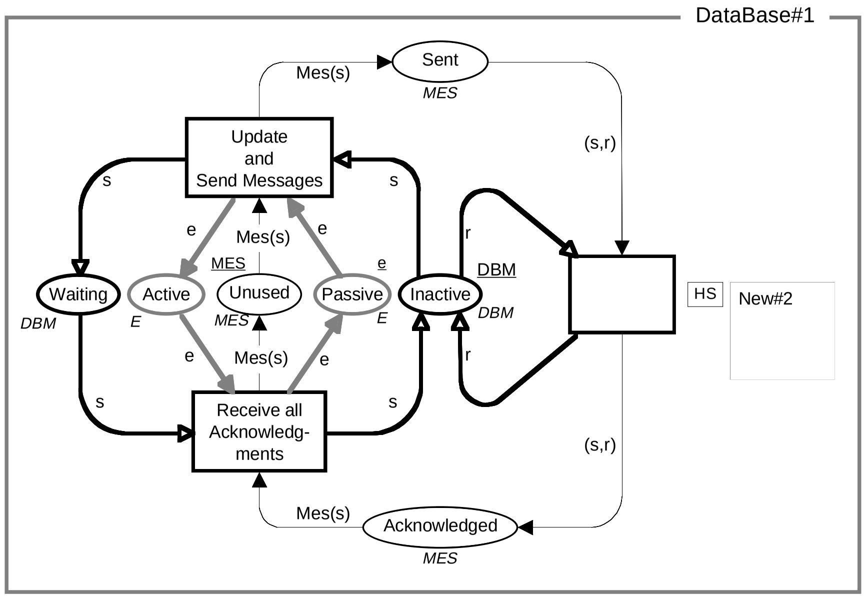
How to make a new subpage

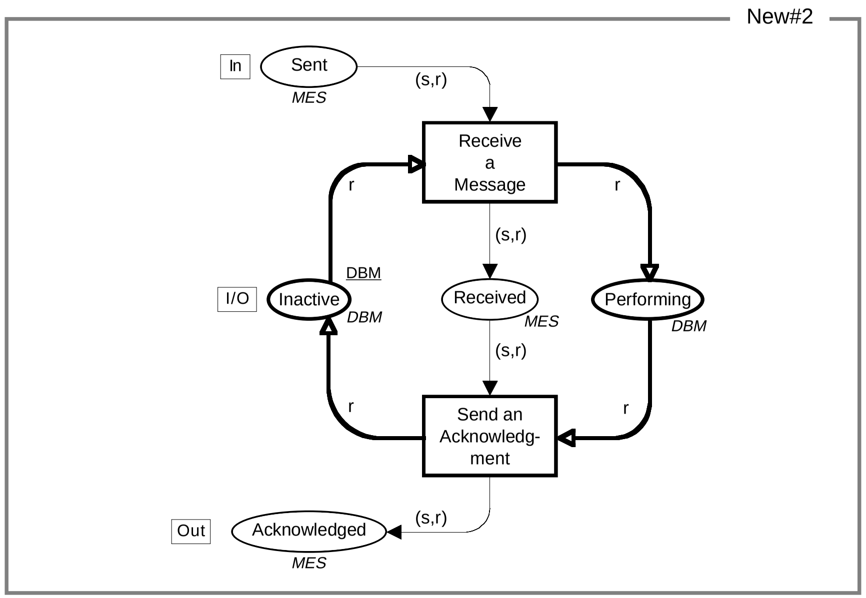


We want to *move* the four selected nodes to a *new page* – and replace them by a *substitution transition:*

- This is done by a single command – called *Move to Subpage*.

Result of Move to Subpage





Move to Subpage is complex

The *Move to Subpage* command is *complex*. The command:

- Checks the *legality of the selection* (all border nodes must be transitions).
- Creates the *new page*.
- *Moves the subnet* to the new page.
- *Prompts the user* to create a new transition which becomes the supernode for the new subpage.
- Creates the *port places* by copying those places which were next to the selected subnet.
- Calculates the *port types* (in, out or in/out).
- Creates the corresponding *port inscriptions*.
- Constructs the necessary *arcs* between the port nodes and the selected subnet.
- Draws the *arcs* surrounding the new transition.
- Creates a *hierarchy inscription* for the new transition.
- Updates the *hierarchy page*.

All these things are done in a *few seconds*.

Top-down and bottom-up

Move to Subpage supports *top-down* development. However, it is also possible to work *bottom-up* – or use a *mixture* of top-down and bottom-up.

The *Substitution Transition* command is used to relate a substitution transition to an *existing page*. The command:

- Makes the *hierarchy page active*.
- *Prompts the user* to select the desired subpage; when the mouse is moved over a page node it blinks, unless it is illegal (because selecting it would make the page hierarchy cyclic).
- *Waits* until a blinking *page node* has been selected.
- Tries to deduce the *port assignment* by means of a set of rules which looks at the port/socket names and the port/socket types.
- Creates the *hierarchy inscription* with the name and number of the subpage and with those parts of the port assignment which could be automatically deduced.
- Updates the *hierarchy page*.

Syntax checking

When a CPN diagram has been constructed it can be *syntax checked*.

The most common errors are:

- Syntax errors in the *declarations*.
- Syntax errors in *arc expressions or guards*.
- *Type mismatch* between arc expressions and colour sets.

Syntax checking is *incremental:*

- When a colour set, guard or an arc expression is changed, it is *sufficient* to recheck the *nearest surroundings*.
- Analogously, if an *arc* is added or removed.

All CPN diagrams in this set of lecture notes are made by means of the CPN editor.

CPN simulator

When a *syntactical correct* CPN diagram has been constructed, the CPN tool generates the necessary *code to perform simulations.*

The simulation code:

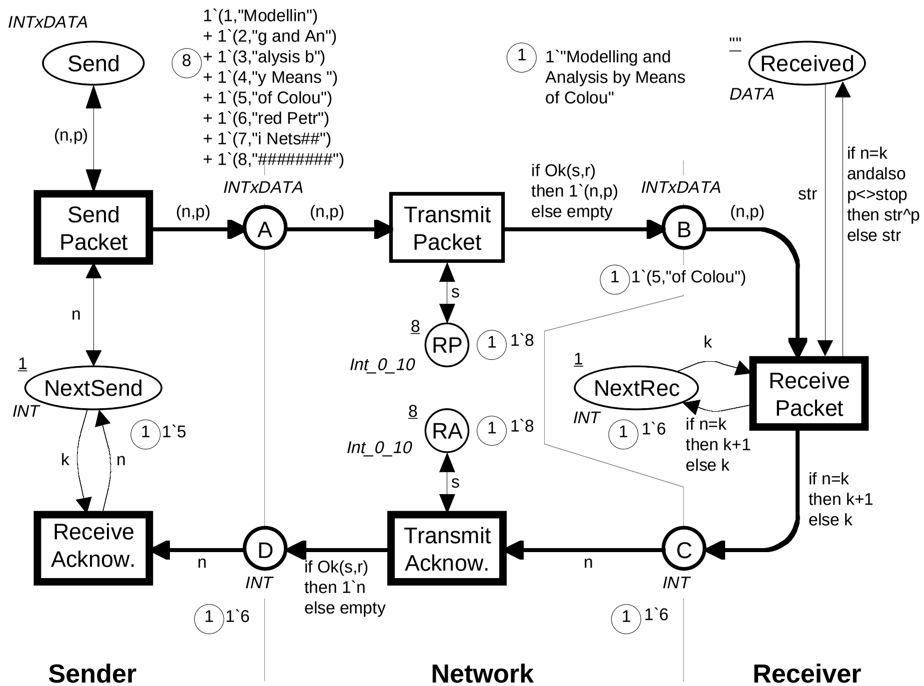
- Calculates whether the individual transitions and bindings are *enabled.*
- Calculates the *effect of occurring transitions and bindings.*

The code generation is *incremental*. Hence it is fast to make small changes to the CPN diagram.

We distinguish between two kinds of simulations:

- In an *interactive* simulation the user is in control, but most of the work is done by the system.
- In an *automatic* simulation the system does all the work.

Interactive simulation



Simulation results are shown directly on the CP-net:

- The user can see the *enabled transitions* and the *markings* of the individual places.

To *execute a step*, the user:

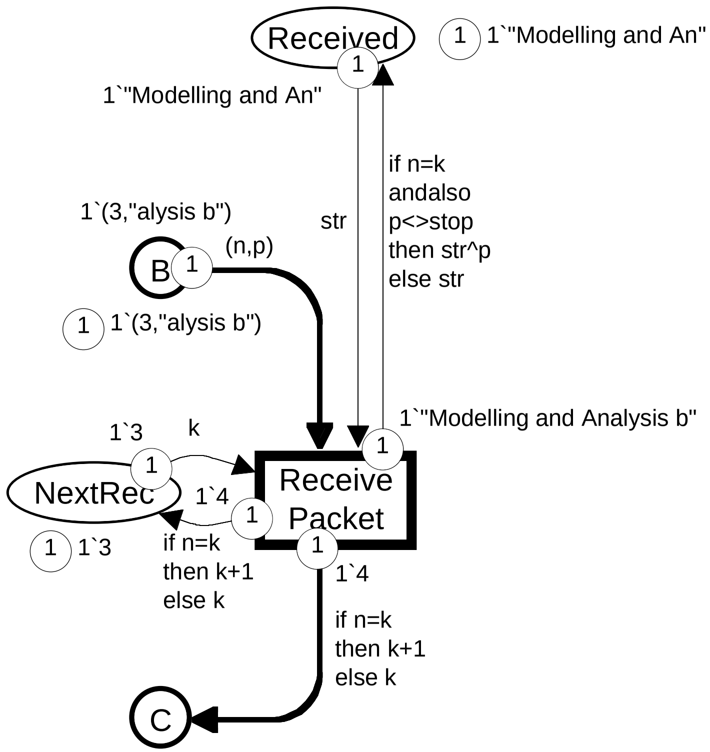
- *Selects* one of the enabled transitions.
- Then he *either* enters a binding or asks the simulator to calculate all the enabled bindings, so that he can select one.

Execution of a step

The simulator:

- Checks the *legality and enabling* of the binding.
- Calculates the *result of the execution.*

The *user determines* whether the simulator displays the tokens which are added/removed:



Interactive simulation with random selection of steps

The simulator *chooses* between conflicting transitions and bindings (by means of a *random number generator*).

- The user can *observe* all details, e.g., the markings the enabling and the added/removed tokens.
- The simulator *shows the page* on which each step is executed – by moving the corresponding window to the top of the screen.
- The user can set *breakpoints* so that he has the necessary time to inspect markings, enablings, etc.

A simulation with this amount of graphical feedback is *slow* (typically a few transitions per minute):

- It takes a lot of time to update the graphics.
- A user has no chance to follow a fast simulation.

It is possible to *turn off* selected parts of the *graphical feedback*, e.g.:

- Added and removed *tokens.*
- Observation of *uninteresting pages.*

Automatic simulation

The simulator *chooses* between conflicting transitions and bindings (by means of a *random number generator*).

The user does *not* intend to follow the simulation:

- The simulation can be *very fast* – several hundred steps per second.
- The user specifies some *stop criteria*, which determine the duration of the simulation.
- When the simulation stops the graphics of the CP-net is *updated*.
- Then the user can inspect all details of the graphics, e.g., the *enabling* and the *marking*.
- Automatic simulations can be *mixed* with interactive simulations.

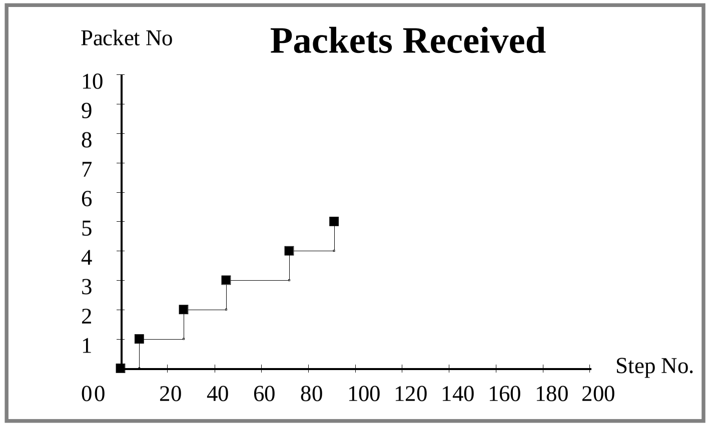
To find out what happens during an *automatic simulation* the user has a number of choices.

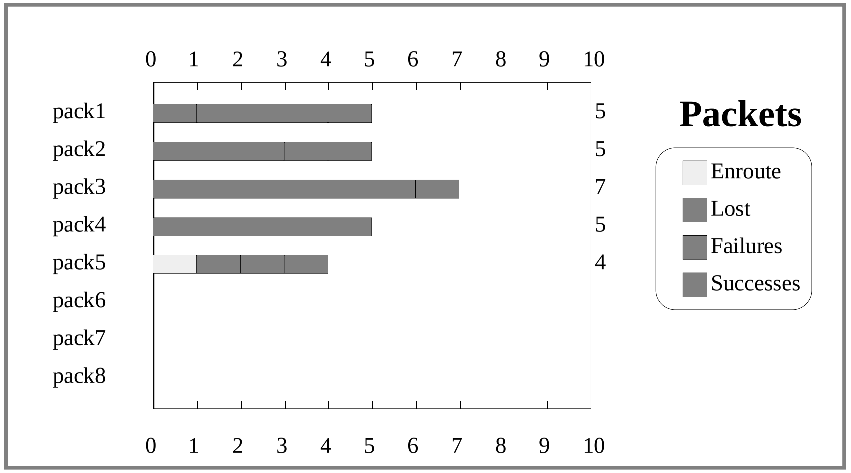
Simulation report

1	SendPack@(1:Top#1)	{n = 1, p = "Modellin"}
2	TranPack@(1:Top#1)	{n = 1, p = "Modellin", r = 6, s = 8}
3	SendPack@(1:Top#1)	{n = 1, p = "Modellin"}
4	TranPack@(1:Top#1)	{n = 1, p = "Modellin", r = 3, s = 8}
5	RecPack@(1:Top#1)	{k = 1, n = 1, p = "Modellin", str = ""}
6	SendPack@(1:Top#1)	{n = 1, p = "Modellin"}
7	SendPack@(1:Top#1)	{n = 1, p = "Modellin"}
8	TranAck@(1:Top#1)	{n = 2, r = 2, s = 8}
9	TranPack@(1:Top#1)	{n = 1, p = "Modellin", r = 7, s = 8}
10	RecPack@(1:Top#1)	{k = 2, n = 1, p = "Modellin", str = "Modellin"}
11	RecAck@(1:Top#1)	{k = 1, n = 2}
12	RecPack@(1:Top#1)	{k = 2, n = 1, p = "Modellin", str = "Modellin"}
13	TranAck@(1:Top#1)	{n = 2, r = 7, s = 8}
14	TranPack@(1:Top#1)	{n = 1, p = "Modellin", r = 6, s = 8}
15	RecAck@(1:Top#1)	{k = 2, n = 2}
16	SendPack@(1:Top#1)	{n = 2, p = "g and An"}
17	TranAck@(1:Top#1)	{n = 2, r = 6, s = 8}
18	RecPack@(1:Top#1)	{k = 2, n = 1, p = "Modellin", str = "Modellin"}
19	RecAck@(1:Top#1)	{k = 2, n = 2}
20	SendPack@(1:Top#1)	{n = 2, p = "g and An"}

The *simulation report* shows the *transitions* which have occurred. The user determines whether he also wants to see the *bindings*.

Charts

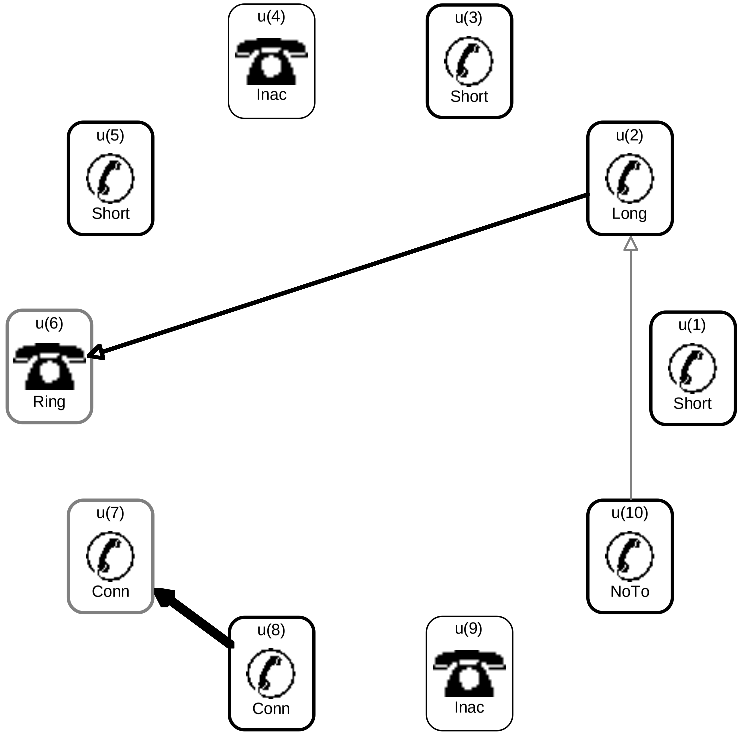




These charts are used to show the *progress* of a simulation of the simple protocol:

- The upper chart is updated each time a new packet is *successfully received.*
- The lower chart is updated for *each 50 steps.*

Other kinds of graphics

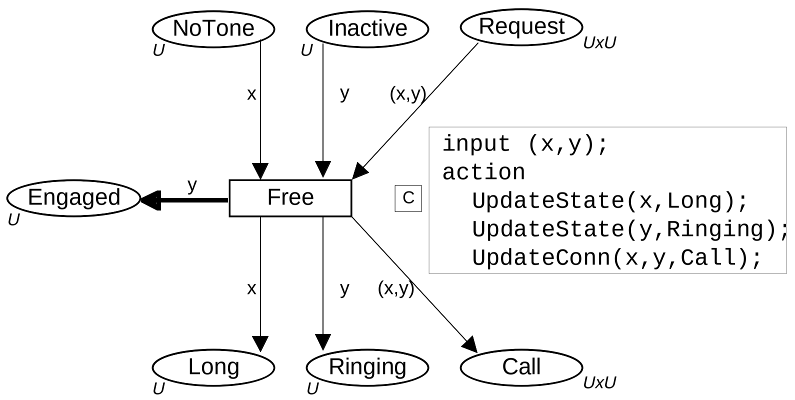


This graphic is used to display the state of a *simple telephone system.* The graphics is updated each time one of the telephones changes to a new state:

- Telephones u(7) and u(8) are *connected.*
- Telephone u(2) is calling u(6) which is *ringing.*
- Telephone u(10) is calling u(2). This call will *not succeed* because u(2) already is engaged.

Code segments

Each transition may have a code segment, i.e., a sequence of *program instructions* which are executed each time the transition occurs.



- The instructions in code segment are used to *update charts and graphics.*
- This is done by calling a number of *library functions.*
- Usually, the code segment does *not* influence the *behaviour* of the CP-net (i.e., the enabling and occurrence).
- However, a code segment may *read and write* from *files.*
- In this way it is possible to *input values* to be used during the simulation, or to *output simulation results.*

Standard ML

Declarations, net inscriptions and code segments are specified in a *programming language* called *Standard ML*.

- *Strongly typed*, *functional* language.
- *Data types* can be:
  - *Atomic* (integers, reals, strings, booleans and enumerations).
  - *Structured* (products, records, unions, lists and subsets).
- Arbitrary complex *functions* and *operations* can be defined (polymorphism and overloading).
- Computational power of expressions are equivalent to *lambda calculus* (and hence to Turing machines).
- Developed at *Edinburgh University* by Robin Milner and his group.
- Standard ML is well-known, well-tested and very general. Several *text books* are available.

Time analysis

CP-nets can be extended with a *time concept*. This means that the *same language* can be used to investigate:

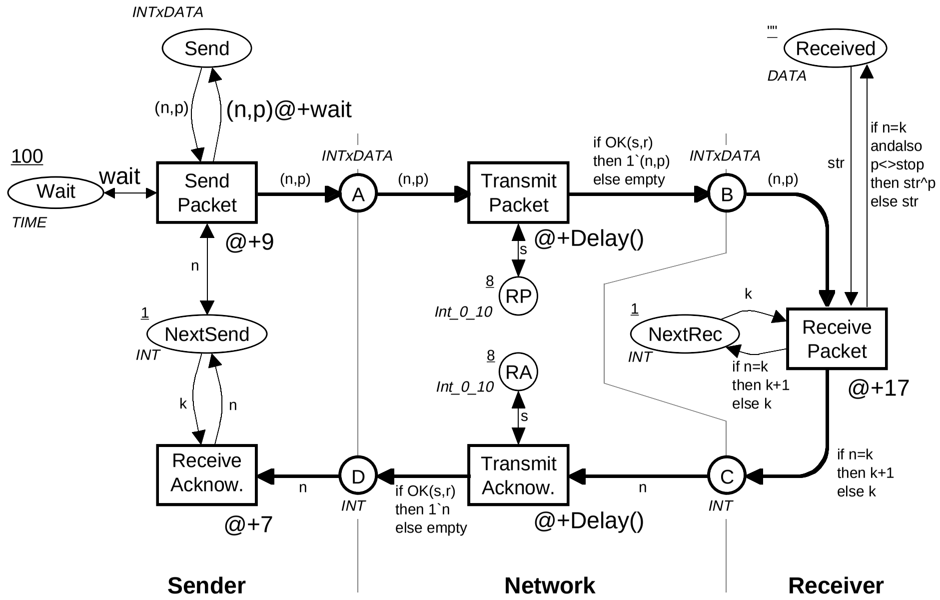
- *Logical correctness*.
  Desired functionality, absence of deadlocks, etc.
- *Performance*.
  Remove bottlenecks. Predict mean waiting times and average throughput. Compare different strategies.

In a timed CP-net each token carries a *colour* (data value) and a *time stamp* (telling when it can be used).

Time stamps are specified by expressions:

- Time stamps can depend upon *colour values*.
- Time stamps can be specified by *probability distributions*.
- This means that we, e.g., can specify *fixed* delays, *interval* delays and *exponential* delays.

A timed CP-net for protocol

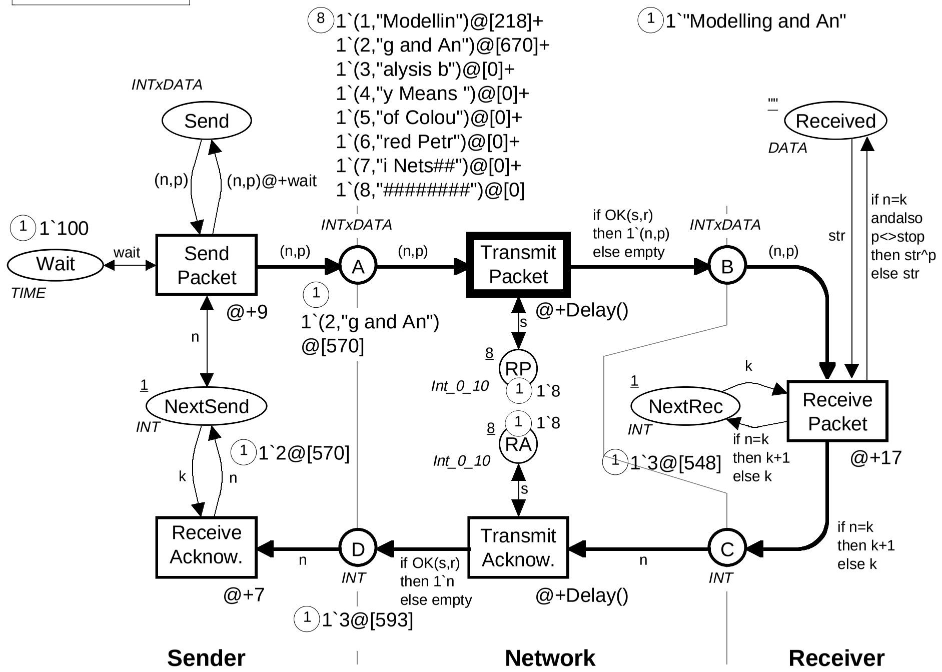


- For the three *Send* and *Receive* operations we specify a *fixed delay*.
- For the *network* we specify an *interval delay*, i.e., random delay between 25 and 75 time units.
- The token colour on place *Wait* specifies the delay between two *retransmissions* of the same packet.

The computer tools for CP-nets also support simulation of *timed* CP-nets.

Timed simulation of protocol

Time: 570



- Model time is now 570.
- *Send Packet* has sent a copy of packet no. 2 at *time 570*.
- If no acknowledgement arrives *another copy* of packet no. 2 will be sent at *time 670*.
- The only transition which is enabled at time 570 is *Transmit Packet*.

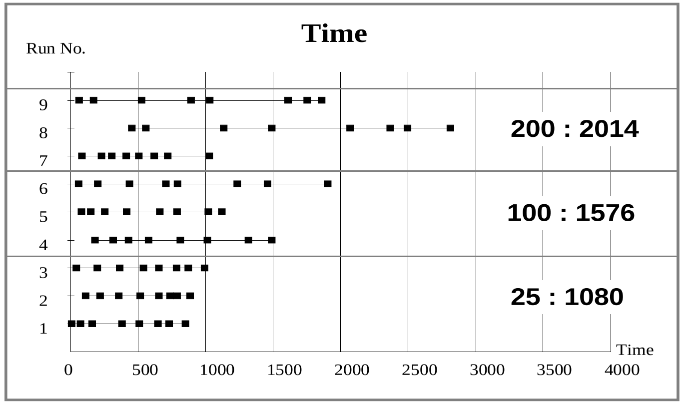
Timed simulations

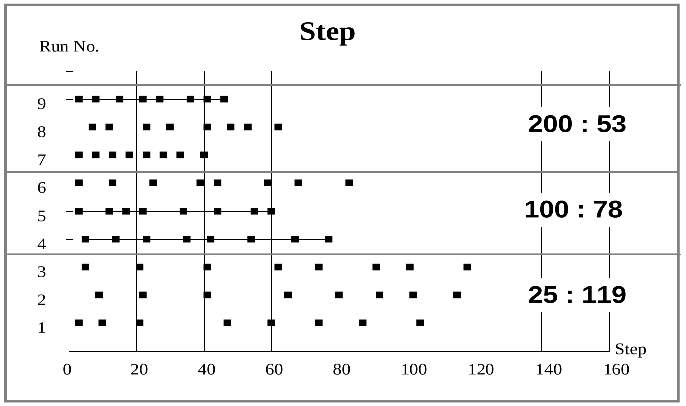
Timed simulations have the *same facilities* as untimed simulations, e.g.:

- We can *switch* between *interactive* and *automatic* simulation.
- *Simulation reports* tell the time at which the individual transitions occurred.
- We can use *charts* and other kinds of *reporting facilities*.

It is easy to *switch* between a *timed* and an *untimed simulation*.

Charts for a timed simulation





- *Short interval* between retransmisions implies *fast transmission* with *heavy use of the network.*
- *Long interval* between retransmisions implies *slow transmission* with *less use of the network.*
- To get *reliable results* it is necessary to make a *large number* of *lengthy* simulation runs.

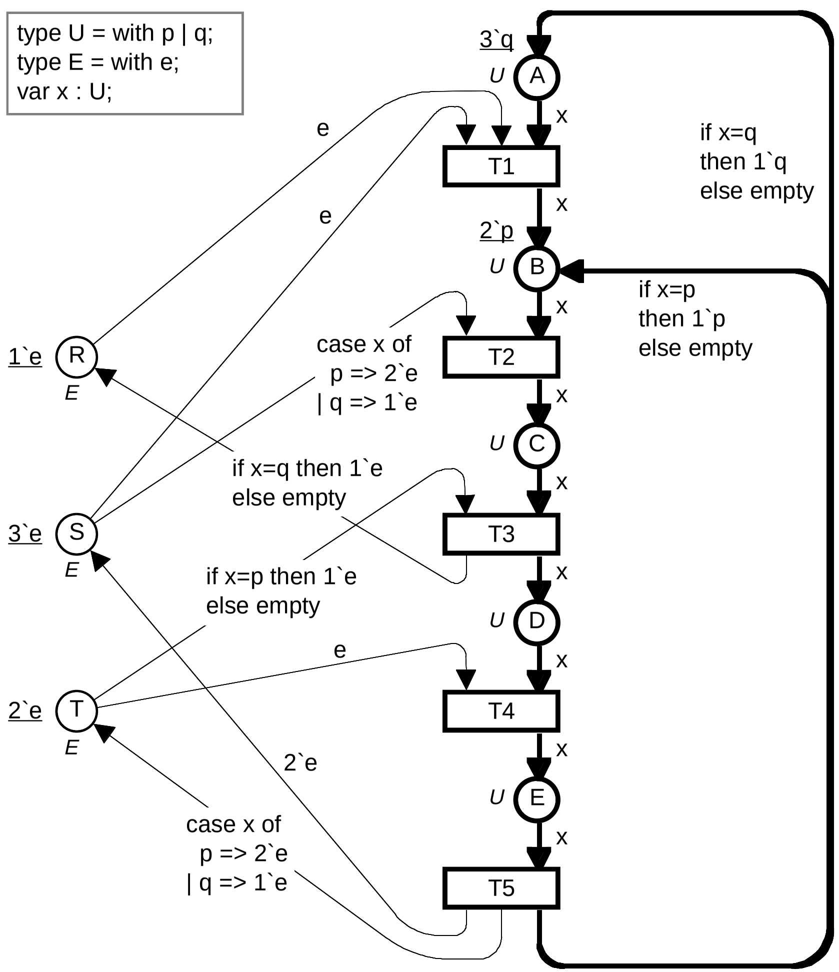
Part 4: Verification of CP-nets

In this part of the talk we describe the two most important methods for *verification* of CP-nets:

- *State spaces* (also called reachability graphs and occurrence graphs).
- *Place invariants*.

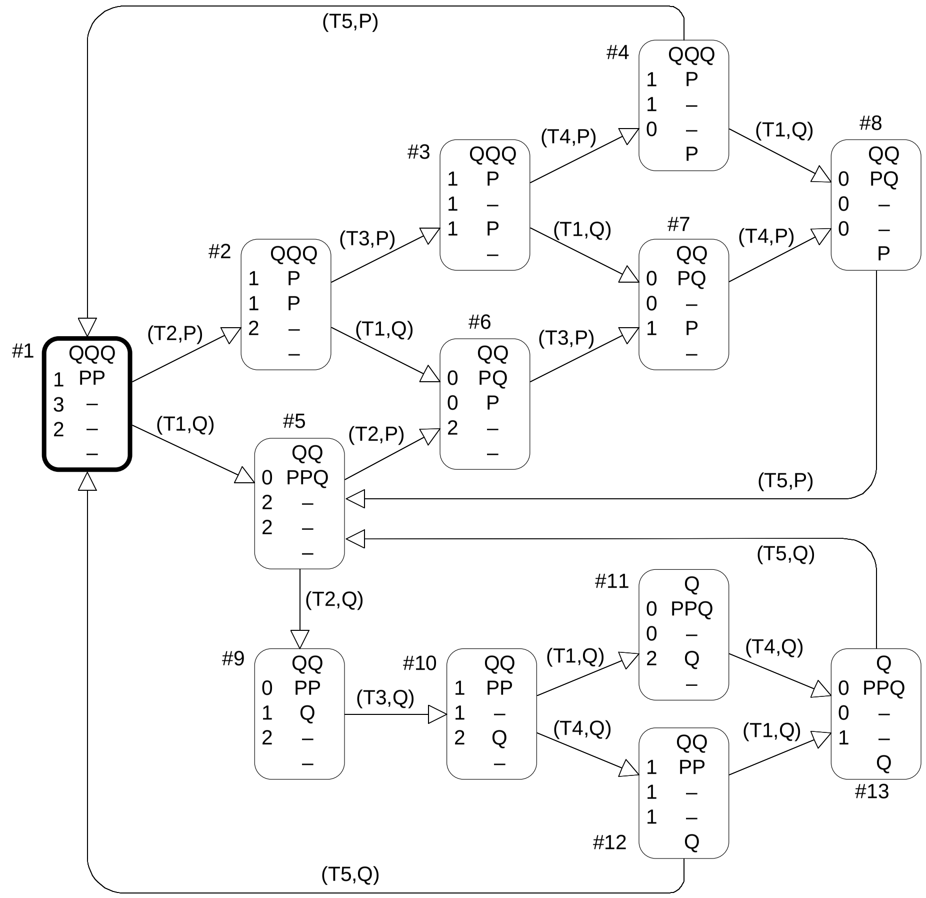
We also describe how the verification methods are supported by *computer tools*.

State space analysis



To obtain a *finite* state space we remove the cycle counters. Otherwise there would be an *infinite* number of reachable markings.

State space for resource allocation



Directed graph with:

- A *node* for each *reachable marking* (i.e., state).
- An *arc* for each *occurring binding element.*

Some questions that can be answered from state spaces

Boundedness properties:

- What is the *maximal number* of tokens on the different places?
- What is the *minimal number* of tokens on the different places?
- What are the *possible token colours*?

Home properties:

- Is it *always* possible to *return* to the initial marking?

Liveness properties:

- Are all transitions live, i.e., can they *always* become enabled *again*?

State space report for resource allocation system

Statistics

Occurrence Graph

Nodes:	13
Arcs:	20
Secs:	1
Status:	Full

Scc Graph

Nodes:	1
Arcs:	0
Secs:	1

Boundedness Properties

Upper Integer Bounds

A:	3
B:	3
C:	1
D:	1
E:	1
R:	1
S:	3
T:	2

Lower Integer Bounds

A:	1
B:	1
C:	0
D:	0
E:	0
R:	0
S:	0
T:	0

Upper Multi-set Bounds

A:	3`q
B:	2`p+ 1`q
C:	1`p+ 1`q
D:	1`p+ 1`q
E:	1`p+ 1`q
R:	1`e
S:	3`e
T:	2`e

Lower Multi-set Bounds

A:	1`q
B:	1`p
C:	empty
D:	empty
E:	empty
R:	empty
S:	empty
T:	empty

State space report (continued)

Home Properties

Home Markings: All

Liveness Properties

Dead Markings: None

Live Transitions: All

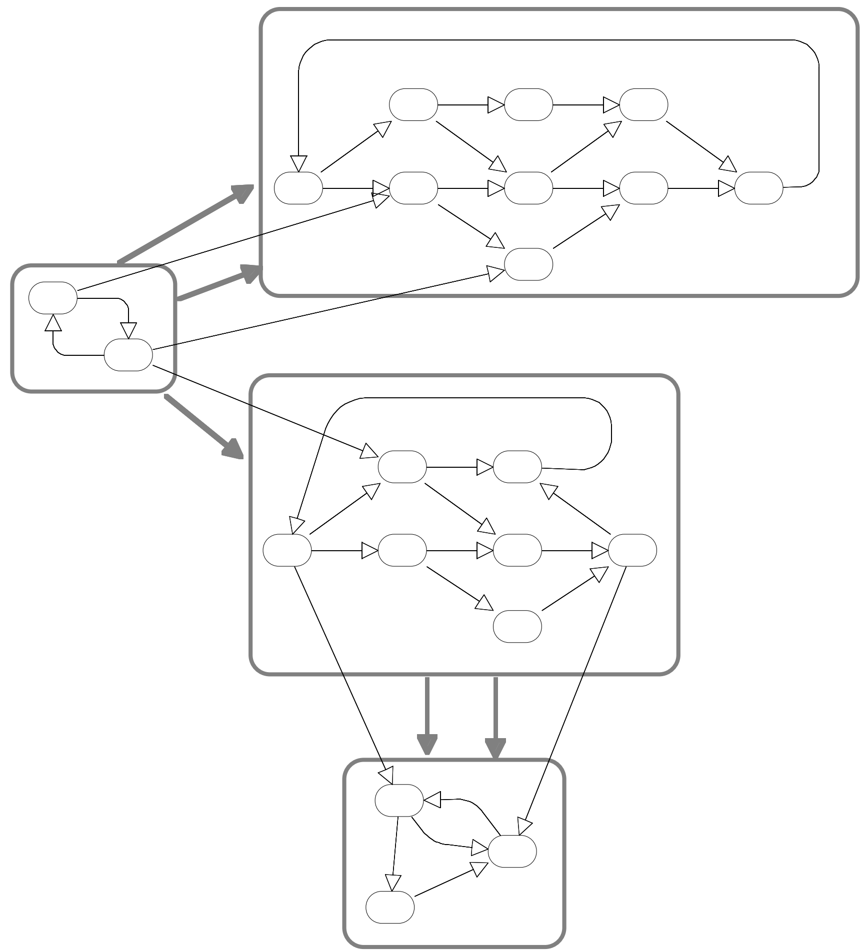
Fairness Properties

T1:	No Fairness
T2:	Impartial
T3:	Impartial
T4:	Impartial
T5:	Impartial

Generation of the state space report takes only a *few seconds*.

- The report contains a lot of *useful information* about the *behaviour* of the CP-net.
- The report is excellent for *locating errors* or to *increase our confidence* in the correctness of the system.

Strongly connected components



- Subgraph where *all nodes are reachable from each other*.
- *Maximal* subgraph with this property.

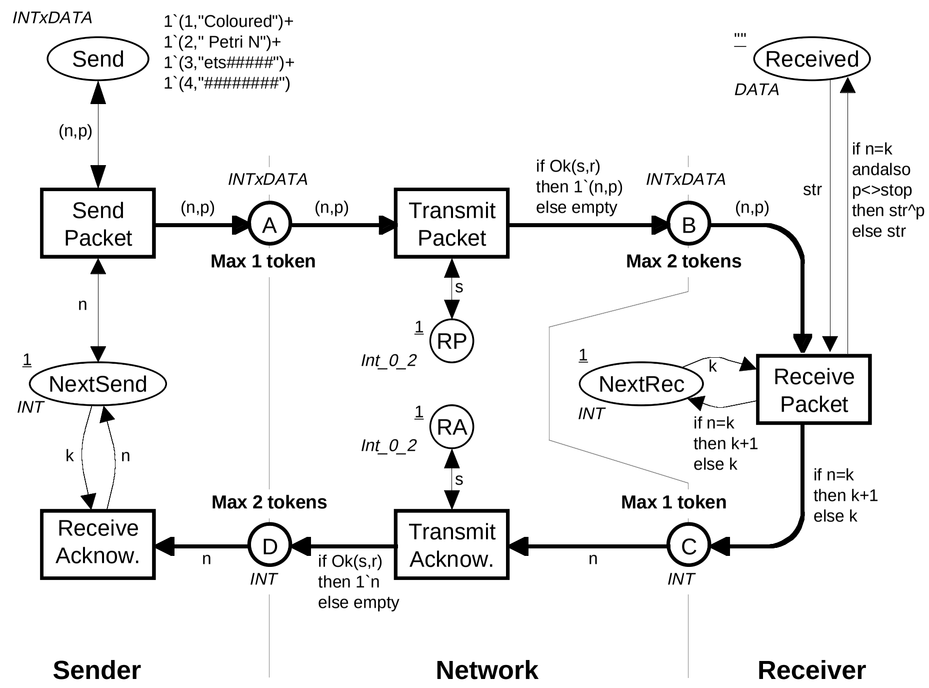
Strongly connected components are very useful

There are often *much fewer* strongly connected components than nodes:

- A *cyclic system* has only *one* strongly connected component.
- This is, e.g., the case for the resource allocation system.
- The *strongly connected components* can be determined in *linear time*, e.g., by Tarjan's algorithm.

Strongly connected components can be used to answer questions about *home properties* and *liveness properties*.

State space for simple protocol



To obtain a *finite* state space we limit the number of tokens on the "buffer" places A, B, C and D. Otherwise there would be an *infinite* number of reachable markings.

Moreover, we now only have *4 packets* and a *binary choice* between success and failure.

State space report for protocol

Statistics

```
Occurrence Graph               Scc Graph
    Nodes:   4298                  Nodes:   2406
    Arcs:    15887                 Arcs:    11677
    Secs:    53                    Secs:    17
    Status:  Full
```

Boundedness Properties

```
Upper Integer Bounds           Lower Integer Bounds
    A:          1                  A:          0
    B:          2                  B:          0
    C:          1                  C:          0
    D:          2                  D:          0

    NextRec:    1                  NextRec:    1
    NextSend:   1                  NextSend:   1

    RA:         1                  RA:         1
    RP:         1                  RP:         1

    Received:   1                  Received:   1
    Send:       4                  Send:       4
```


State space report (continued)

```
Upper Multi-set Bounds
    A:          1`(1,"Coloured")+ 1`(2," Petri N")+
                1`(3,"ets#####")+ 1`(4,"########")
    B:          2`(1,"Coloured")+ 2`(2," Petri N")+
                2`(3,"ets#####")+ 2`(4,"########")
    C:          1`2+ 1`3+ 1`4+ 1`5
    D:          2`2+ 2`3+ 2`4+ 2`5

    NextRec:    1`1+1`2+1`3+1`4+1`5
    NextSend:   1`1+1`2+1`3+1`4+1`5

    RA:         1`1
    RP:         1`1

    Received:   1`""+ 1`"Coloured"+ 1`"Coloured Petri N"+
                1`"Coloured Petri Nets#####"
    Send:       1`(1,"Coloured")+ 1`(2," Petri N")+
                1`(3,"ets#####")+ 1`(4,"########")

Lower Multi-set Bounds
    A:          empty
    B:          empty
    C:          empty
    D:          empty

    NextRec:    empty
    NextSend:   empty

    RA:         1`1
    RP:         1`1

    Received:   empty
    Send:       1`(1,"Coloured")+ 1`(2," Petri N")+
                1`(3,"ets#####")+ 1`(4,"########")
```

State space report (continued)

Home Properties

Home Markings:	1	[452]

Liveness Properties

Dead Markings:	1	[452]
Live Transitions:	None	

Fairness Properties

Send Packet:	Impartial
Transmit Packet:	Impartial
Receive Packet:	No Fairness
Transmit Acknow:	No Fairness
Receive Acknow:	No Fairness

Generation of the state space report takes only a *few seconds*.

- The report contains a lot of *useful information* about the *behaviour* of the CP-net.
- The report is excellent for *locating errors* or to *increase our confidence* in the correctness of the system.

Investigation of dead marking

We ask the system to display marking number 452.

```
452
NextSend = 5
NextRec = 5
Received = "Coloured Petri Nets#####"
```

```
452
8:0
```

Marking no. 452 is the *desired final marking* (all packets has been received in the correct order)

Marking 452 is *dead:*

- This implies that the protocol is *partially correct* (if execution stops it stops in the desired final marking).

Marking 452 is a *home marking:*

- This implies that we *always have a chance to finish correctly* (it is impossible to reach a state from which we cannot reach the desired final marking).

Investigation of shortest path

We ask the system to calculate one of the *shortest paths* from the initial marking to the dead marking:

```
val path =
NodesInPath(1,452);
```

```
> val path =
[1,2,3,5,8,11,15,20,27,38,50,
64,80,102,133,164,199,243,
301,375,452] : Node list
```

```
Length(path);
```

```
> 20 : int
```

The calculated path contains *20 transitions.*

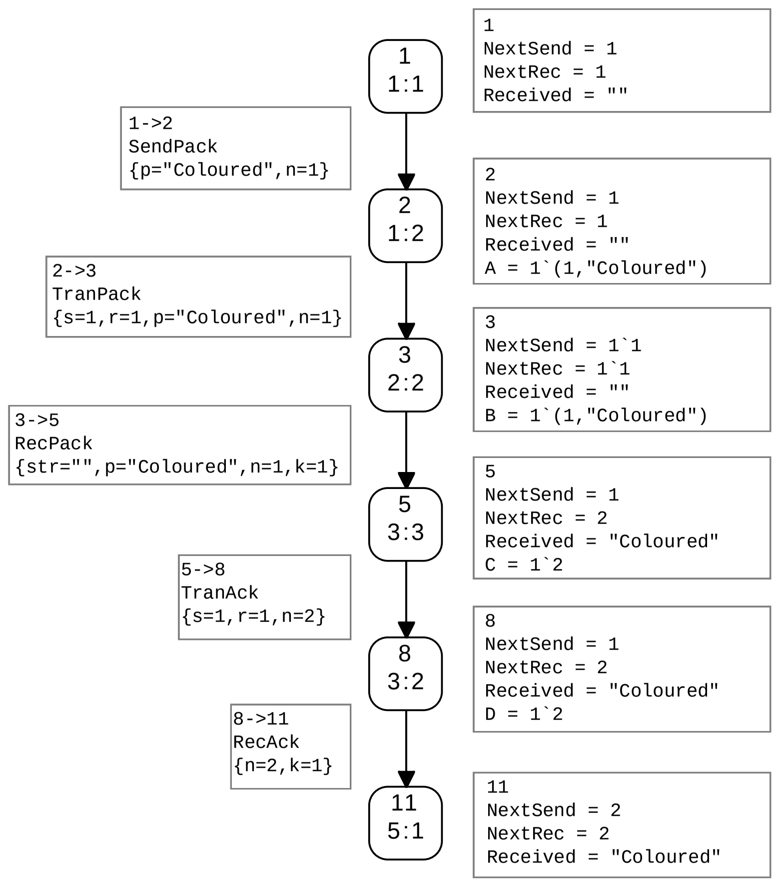
- This is as expected because there are *4 packets* which each needs *5 transitions* to occur.

Drawing of shortest path

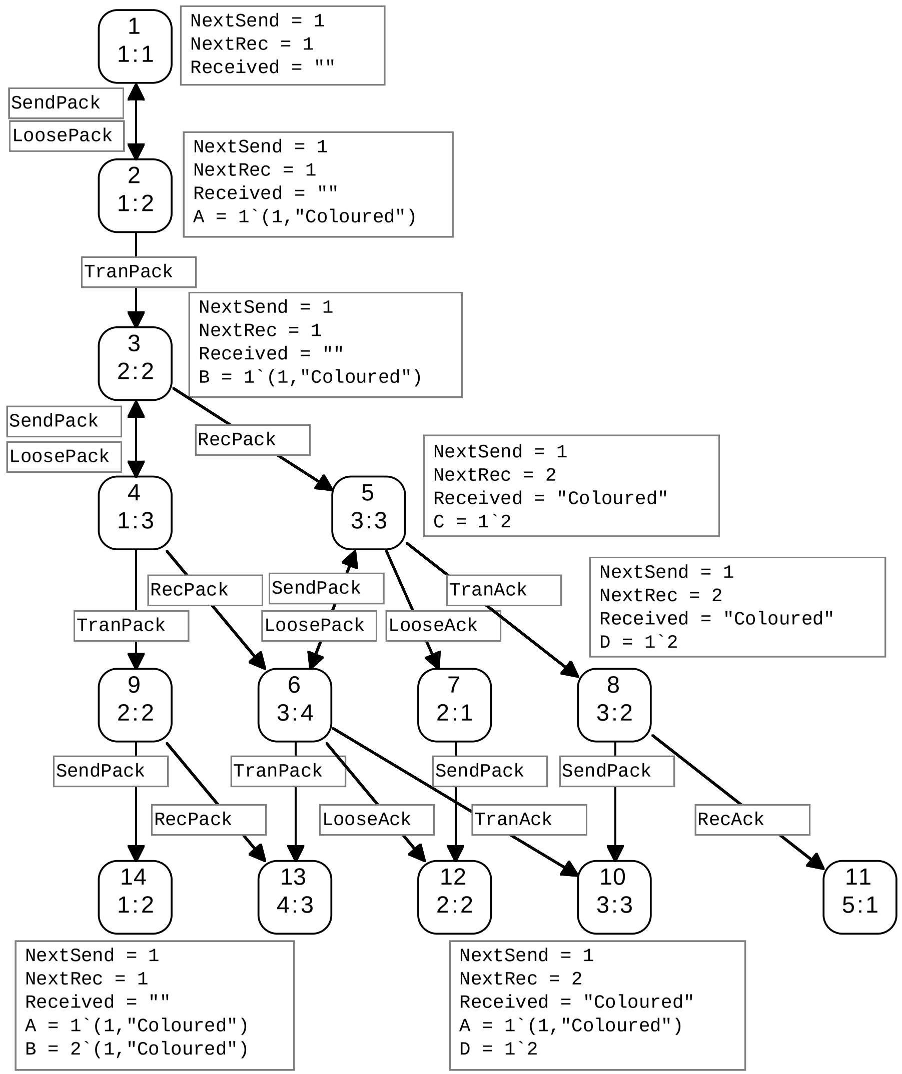
We ask the system to draw the *first six nodes* in the calculated shortest path:

```
DisplayNodePath; [1,2,3,5,8,11];
```

```
> () : unit
```



Draw subgraph

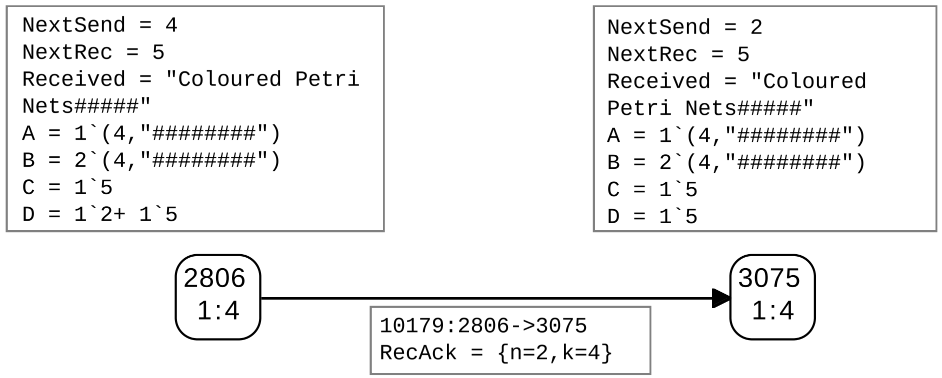


Non-standard questions

We ask the system to search *all arcs* in the *entire graph* and return the *first 10 arcs* where *NextSend* has a *larger* value in the *source marking* than it has in the *destination marking*.

```
PredArcs
      (EntireGraph,
       fn a => ((ms_to_col(Mark.NextSend 1
                  (SourceNode a))) >
                (ms_to_col(Mark.NextSend 1
                  (DestNode a)))),
       10)
end;
```

```
>[10179,10167,10165,10159,10055,10052,10035,
10031,10019,10007] : Arc list
```



Temporal logic

It is also possible to make questions by means of a CTL-like *temporal logic.*

Usually CTL focuses on queries about *state properties*, e.g.:

- Inv(Pos(M))
  checks whether M is a *home marking.*
- Ev(dead)
  checks whether there are any infinite occurrence sequences.

Our version of CTL also allows queries about *transitions* and *binding elements.*

- Inv(Pos(t in Arc))
  checks whether transition t is *live*.

Timed CP-nets

The computer tools for CP-nets also support state space analysis of *timed* CP-nets.

State spaces – pro/contra

State spaces are *powerful* and *easy* to use.

- The main drawback is the *state explosion, i.e., the size of the state space*.
- The present version of our tool handles graphs with 250,000 nodes and 1,000,000 arcs. For many systems this is *not sufficient.*

Fortunately, it is sometimes possible to construct much more *compact* state spaces – *without loosing information.*

- This is done by exploiting the inherent *symmetries* of the modelled system.
- We define two *equivalence relations* (one for markings and one for binding elements).
- The condensed state spaces are often *much smaller* (polynomial size instead of exponential).
- The condensed state spaces contain the *same information* as the full state spaces.

Place invariants analysis

The basic idea is similar to the use of *invariants* in *program verification*.

- A place invariant is an *expression* which is satisfied for all reachable markings.
- The expression *counts* the tokens of the marking – using a specified set of weights.

We first *construct* a set of place invariants.

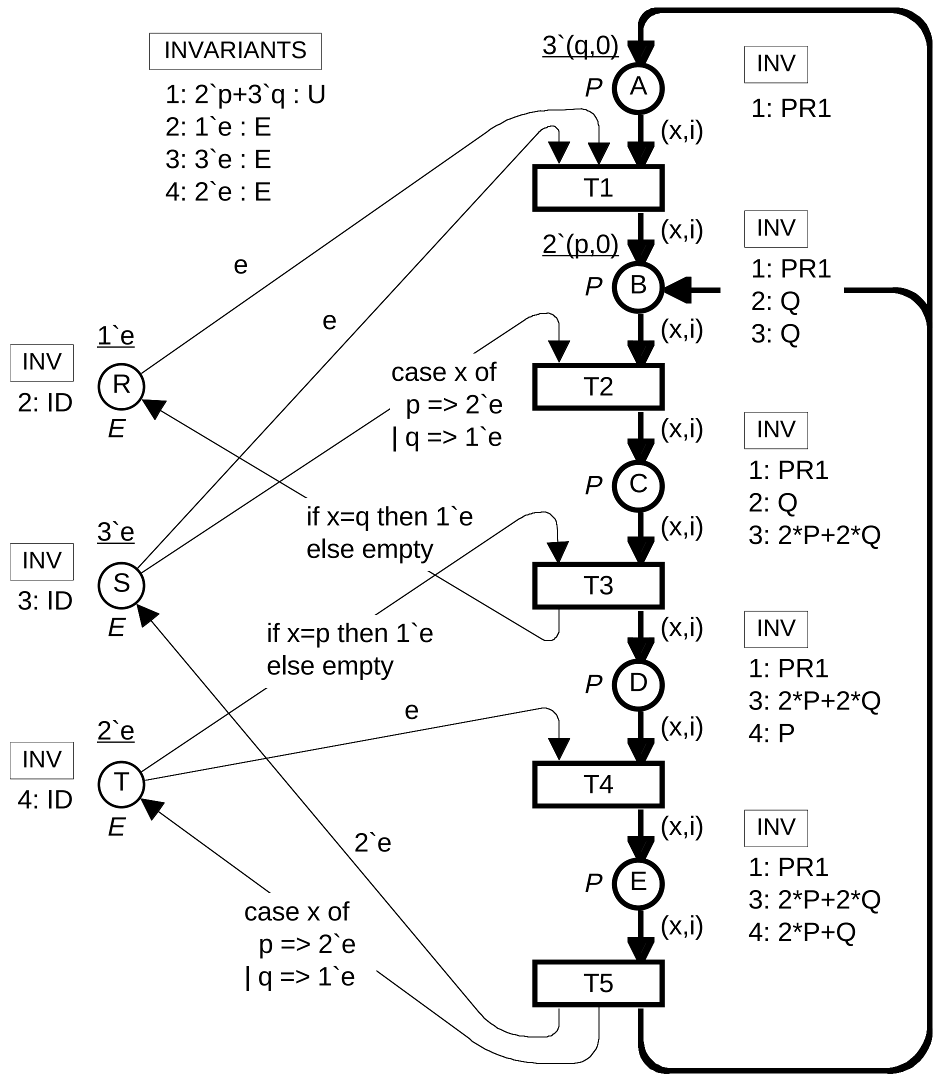
Then we check whether they are *fulfilled*.

- This is done by showing that each occurring binding element *respects* the invariants.
- The *removed* set of tokens must be identical to the *added* set of tokens – when the weights are taken into account.

Finally, we use the place invariants to *prove* behavioural properties of the CP-net.

- This is done by a *mathematical proof*.

Example of place invariants



Place invariants for resource allocation system

To specify the weights we use *three functions:*

- PR_1 is a *projection* function: (x,i) --> x.
- P is an *indicator* function: (p,i) --> 1`e; (q,i) -->Ø.
- Q is an *indicator* function: (p,i) --> Ø; (q,i) -->1`e.
- P and Q “counts” the number of p and q tokens.

PR_1(M(A)+M(B)+M(C)+M(D)+M(E)) = 2`p+3`q

M(R) + Q(M(B)+M(C)) = 1`e

M(S) + Q(M(B)) +
(2*P+2*Q)(M(C)+M(D)+M(E)) = 3`e

M(T) + P(M(D)) + (2*P+Q)(M(E)) = 2`e

A more readable version of the place invariants

PR_1(A+B+C+D+E) = 2`p+3`q

R + Q(B+C) = 1`e

S + Q(B) + (2*P+2*Q)(C+D+E) = 3`e

T + P(D) + (2*P+Q)(E) = 2`e

The place invariants can be used to *prove* properties of the resource allocation system, e.g., that it is *impossible to reach a dead marking.*

Tool support for place invariants

Check of place invaritans:

- The *user* proposes a set of weights.
- The *tool* checks whether the weights constitute a place invariant.

Automatic calculation of all place invariants:

- This is possible, but it is a very *complex* task.
- Moreover, it is difficult to represent the results on a *useful form*, i.e., a form which can be used by the system designer.

Interactive calculation of place invaritans:

- The *user* proposes some of the weights.
- The *tool* calculates the *remaining weights* – if possible.

Interactive calculation of place invariants is *much easier* than a fully automatic calculation.

How to use place invariants

Invariants in ordinary *programming languages:*

- No one would construct a large program – and then expect *afterwards* to be able to calculate invariants.
- Instead invariants are constructed *together* with the program.

For *CP-nets* we should do the same:

- During the system specification and modelling the designer gets a lot of *knowledge* about the system.
- Some of this knowledge can easily be formulated as *place invariants.*
- The invariants can be *checked* and in this way it is possible to find *errors*.
- It can be seen *where* the errors are.

Some *prototypes* of computer tools for invariants analysis do exist. However, none of them are at a state where they can be widely used.

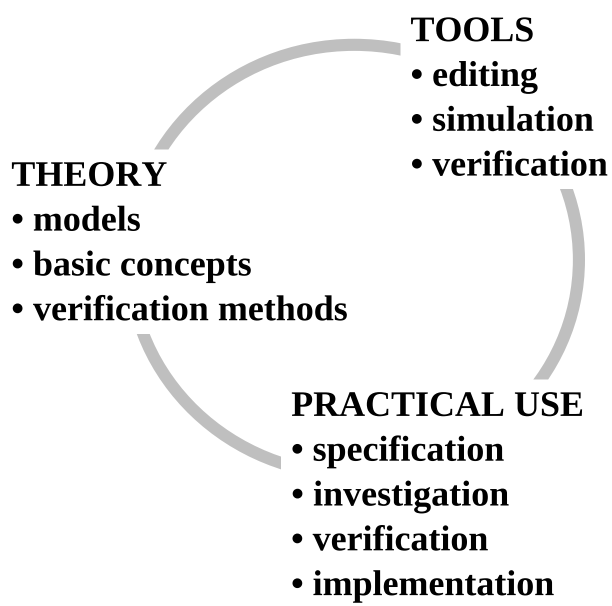
Place invariants – pro/contra

From place invariants it is possible to prove many kinds of *behavioural properties*.

- Invariants can be used to make *modular verification* – because it is possible to combine invariants of the individual pages.
- Invariants can be used to verify *large systems* – without computational problems.
- The user needs some ingenuity to *construct* invariants. This can be supported by *computer tools* – interactive process.
- The user also needs some ingenuity to *use* invariants. This can also be supported by *computer tools* – interactive process.
- Invariants can be used to verify a system – without fixing the *system parameters* (such as the number of sites in the data base system).

Conclusion

One of the main reasons for the success of CP-nets is the fact that we – *simultaneously* – have worked with:



More information on CP-nets

The following WWW pages contain a lot of information about CP-nets and their computer tools:

http://www.daimi.au.dk/CPnets/

A detailed introduction to CP-nets can be found in the following papers/books:

L.M. Kristensen, S. Christensen and K. Jensen: *The Practitioner's Guide to Coloured Petri Nets*. Int. Journal on Software Tools for Technology Transfer, 2 (1998), Springer Verlag, 95-191

K. Jensen: *An Introduction to the Theoretical Aspects of Coloured Petri Nets*. In: J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.): A Decade of Concurrency, Lecture Notes in Computer Science vol. 803, Springer-Verlag 1994, 230-272.

K. Jensen: *An Introduction to the Practical Use of Coloured Petri Nets*. In: W. Reisig and G. Rozenberg (eds.): Lectures on Petri Nets II: Applications, Lecture Notes in Compute Science Vol. 1492, Springer-Verlag 1998, 237-292.

K. Jensen: *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use*. Monographs in Theoretical Computer Science, Springer-Verlag.

- Vol. 1: Basic Concepts, 1992, ISBN: 3-540-60943-1.
- Vol. 2: Analysis Methods, 1994, ISBN: 3-540-58276-2.
- Vol. 3: Practical Use, 1997, ISBN: 3-540-62867-3.

Some of the most important papers on high-level nets, their verification methods and applications have been reprinted in:

K. Jensen, G. Rozenberg (eds.): *High-level Petri Nets. Theory and Application.* Springer-Verlag, 1991, ISBN: 3-540-54125-X.

A list of papers that describe industrial use of CP-nets and their tools can be found on:

http://www.daimi.au.dk/CPnets/intro/example_indu.html

1)

BEHAVIOUR OF ELEMENTARY NET SYSTEMS

G. ROZENBERG

2)

EN SYSTEM

$\mathcal{N} = (B, E, F, C_{in})$

(B, E, F) → underlying static structure } net — potential dynamics

C_{in} → initial state (dynamic state space) — actual dynamics

$\mathcal{C}_{\mathcal{N}}$

$\mathcal{U}_{\mathcal{N}}$

3.

ALL IS FINITE !

EN SYSTEMS ARE

CONTACT-FREE

4)

WHAT IS THE

BEHAVIOUR ?

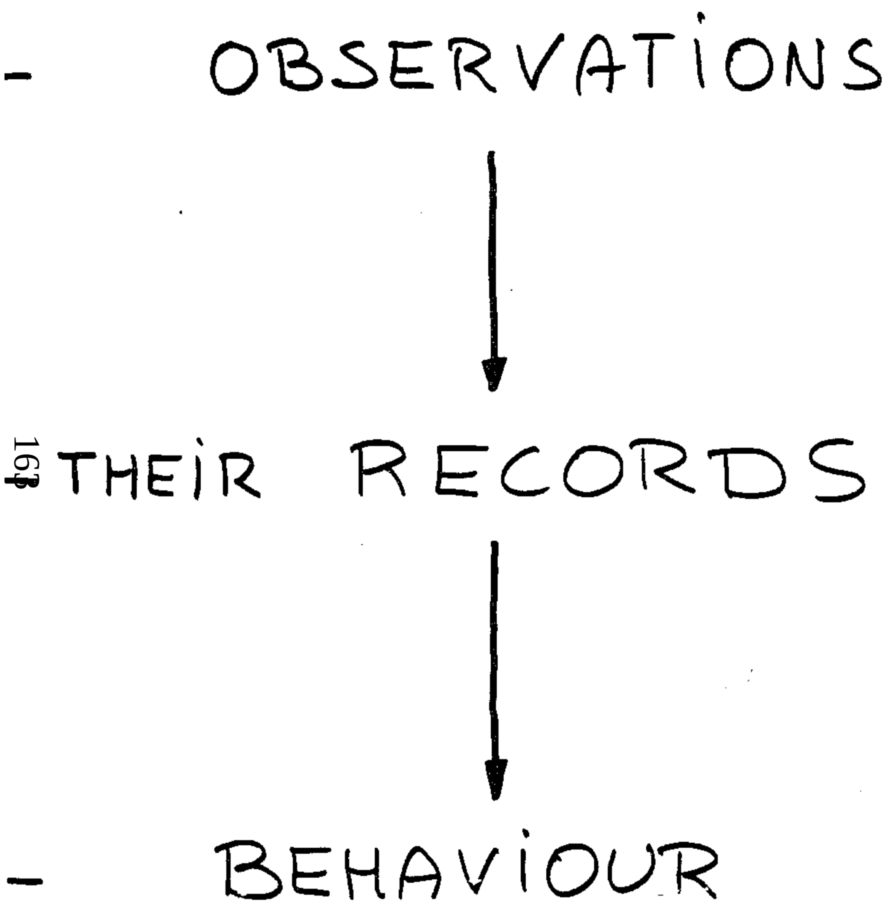
⇕

HOW CAN THE

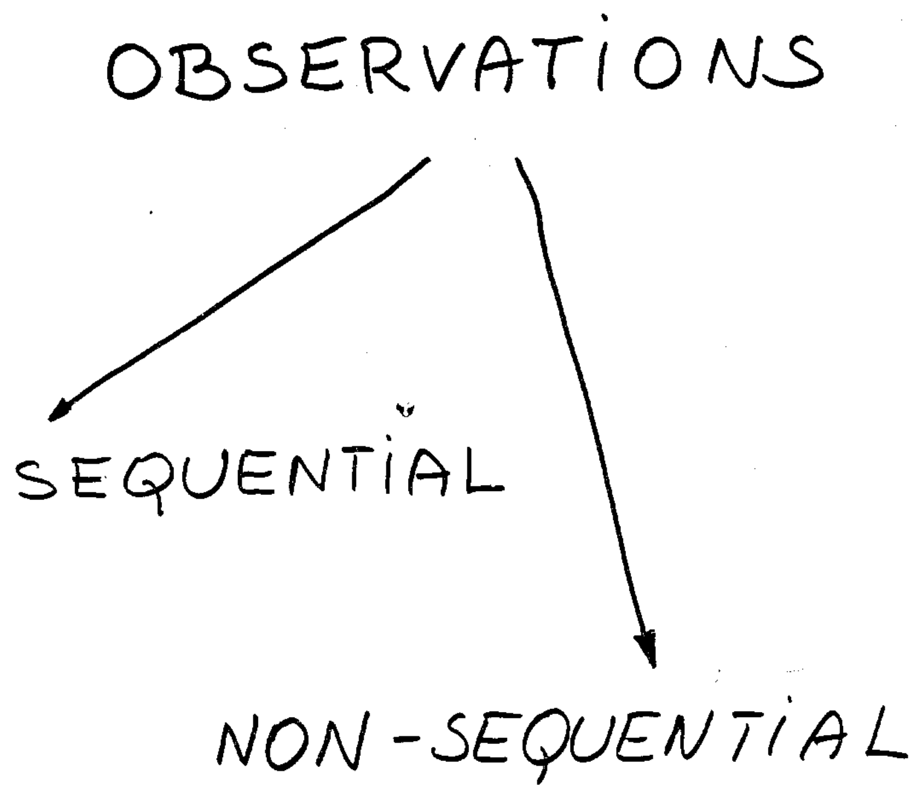
BEHAVIOUR

BE OBSERVED ?

5)

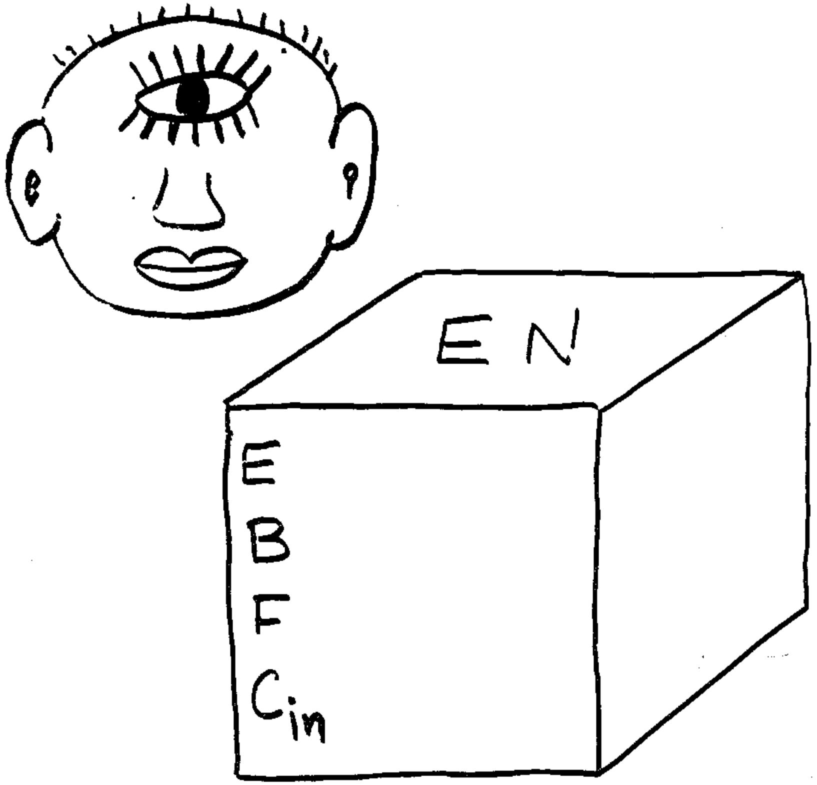

- OBSERVATIONS
- THEIR RECORDS
- BEHAVIOUR

6)


OBSERVATIONS
SEQUENTIAL
NON-SEQUENTIAL

7)

SEQUENTIAL OBSERVATIONS



OBSERVED ARE:

OCCURRENCES OF **SINGLE** EVENTS

8)

$\varrho \in E^*$ IS A FIRING SEQUENCE

IFF

$\varrho = \Lambda$

OR

$\varrho = e_1 \ldots e_n \quad n \geq 1$

$e_1, \ldots, e_n \in E$

WHERE

$(\exists\, C_0, C_1, \ldots, C_n \in \mathcal{C}_N)$

$$C_0 [e_1\rangle C_1 [e_2\rangle \ldots [e_n\rangle C_n$$

$$\|$$

$$C_{in}$$

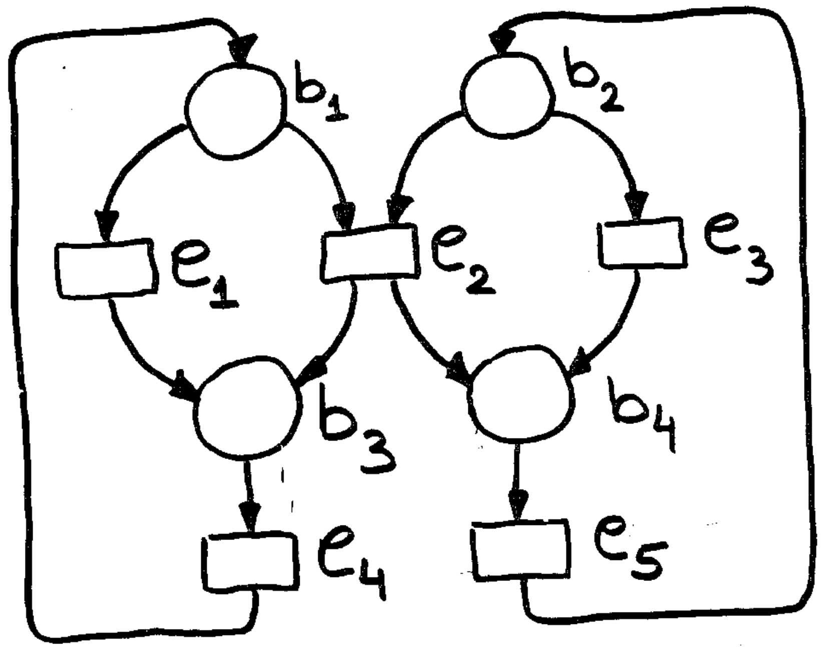
FS($\mathcal{N}$)

TO ANALYZE **FS**($\mathcal{N}$) WE NEED THE SEQUENTIAL CASE GRAPH OF $\mathcal{N}$

SCG($\mathcal{N}$)

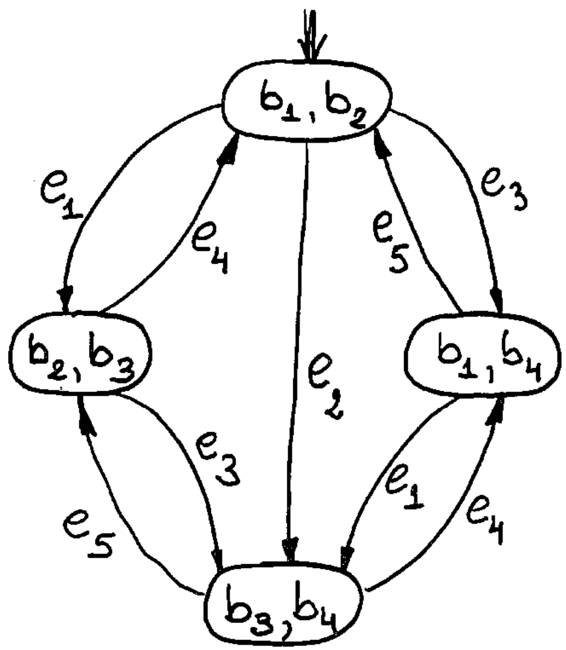
OBTAINED BY DELETING FROM CG($\mathcal{N}$) ALL EDGES *LABELED BY* NON-SINGLETON STEPS

$\mathcal{N}_1$:



$C_{in} = \{b_1, b_2\}$

11)



SCG($\mathcal{N}_1$)

12)

FS($\mathcal{N}_1$):

$e_1\ e_4\ e_2\ e_5\ e_3 \quad \in$

$e_1\ e_3\ e_5\ e_4\ e_1 \quad \in$

$e_1\ e_2\ e_4 \quad \notin$

13)

(1) $\mathbf{FS}(\mathcal{N})$ IS PREFIX CLOSED

$$\left.\begin{array}{l}\rho \in \mathbf{FS}(\mathcal{N}) \\ \rho = \rho_1 \rho_2\end{array}\right\} \rho_1 \in \mathbf{FS}(\mathcal{N})$$

(2) $SCG(\mathcal{N})$ IS FINITE

<u>THEOREM</u>

$(\forall \mathcal{N})_{EN}$ [$\mathbf{FS}(\mathcal{N})$ IS A PREFIX CLOSED REGULAR LANGUAGE] ∎

14)

PROBLEMS !!!

$$\ldots\ldots e_1 e_3 \ldots\ldots \in$$

IS IT A CAUSAL ORDER?

IS IT ONLY AN OBSERVATIONAL ORDER?

IS $\{e_1, e_3\}$ A STEP?

15)

HOW TO EXTRACT (RECOVER) <u>CAUSAL</u> <u>ORDERS</u> FROM SEQUENTIAL OBSERVATIONS ?

THEORY OF TRACES

(MAZURKIEWICZ)

(1)

16)

$\mathcal{N}$ EN SYSTEM

THE INDEPENDENCE RELATION INDUCED BY $\mathcal{N}$: $I_{\mathcal{N}}$

$$(\forall e_1, e_2 \in E_{\mathcal{N}})$$
$$[(e_1, e_2) \in I_{\mathcal{N}} \quad \text{IFF} \quad ({}^{\bullet}e_1 \cup e_1^{\bullet}) \cap ({}^{\bullet}e_2 \cup e_2^{\bullet}) = \emptyset\,]$$

.....

THE DEPENDENCE RELATION INDUCED BY $\mathcal{N}$: $D_{\mathcal{N}}$

$$D_{\mathcal{N}} = (E_{\mathcal{N}} \times E_{\mathcal{N}}) - I_{\mathcal{N}}$$

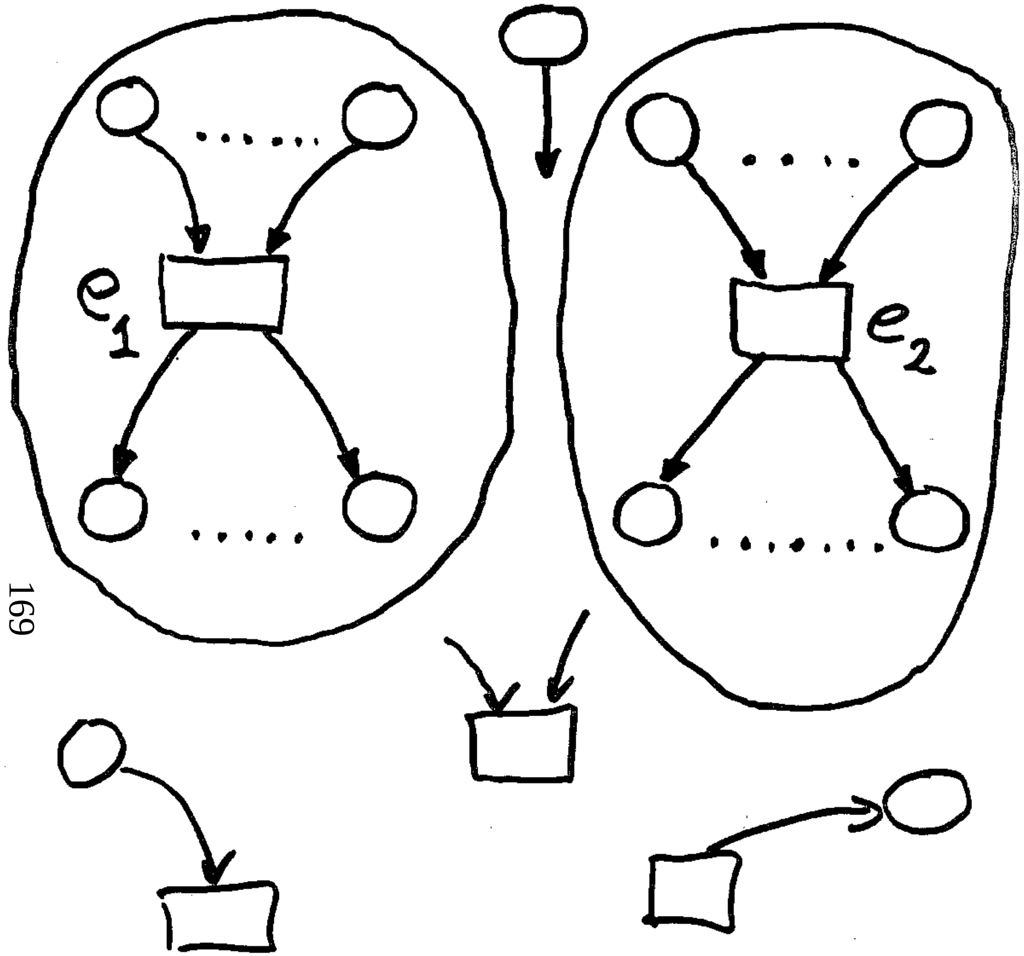
....

$I_{\mathcal{N}}$ is SYMMETRIC & IRREFLEXIVE

$D_{\mathcal{N}}$ is SYMMETRIC & REFLEXIVE

②

17)



$(e_1, e_2) \in I_{\mathcal{N}}$

18)

③

$\rho = \ldots\ldots e_1 e_2 \ldots\ldots \in$ ~~FS~~ $E^*(\mathcal{N})$

$\mu = \ldots\ldots e_2 e_1 \ldots\ldots$

$$\frac{(e_1, e_2) \in I_{\mathcal{N}}}{\rho \doteq_{I_{\mathcal{N}}} \mu}$$

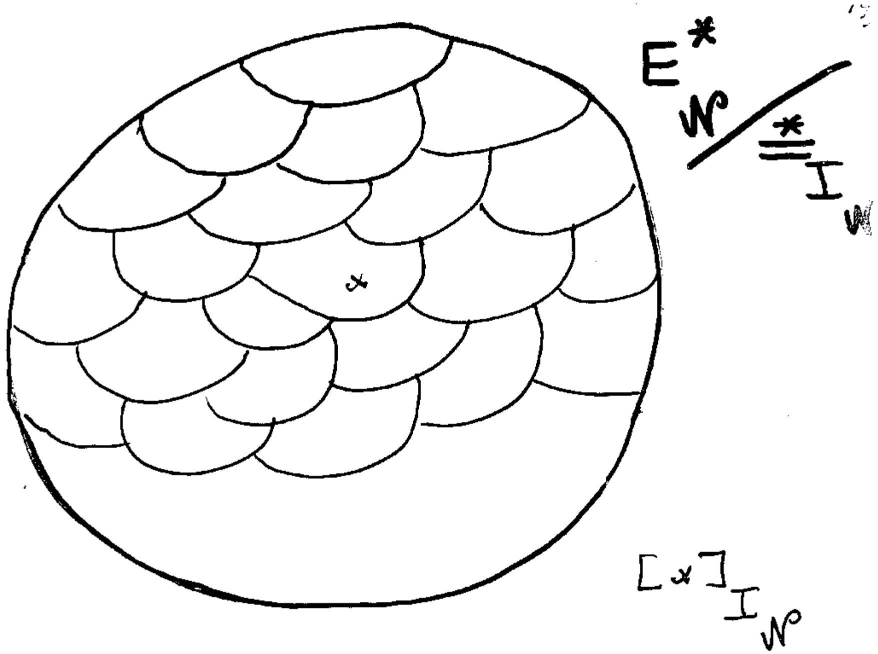
- - - - - - - - - - - - - - - -

$$\frac{\rho \doteq_{I_{\mathcal{N}}} \mu \doteq_{I_{\mathcal{N}}} \gamma \ldots \doteq_{I_{\mathcal{N}}} \delta}{\rho \overset{*}{=}_{I_{\mathcal{N}}} \delta}$$

- - - - - - - - - - - - - - - -

$\overset{*}{=}_{I_{\mathcal{N}}}$ an equivalence relation on ~~FS~~ $E^*(\mathcal{N})$

4



$Z \subseteq E^*_{\mathcal{W}}$ is **$I_{\mathcal{W}}$-CONSISTENT**

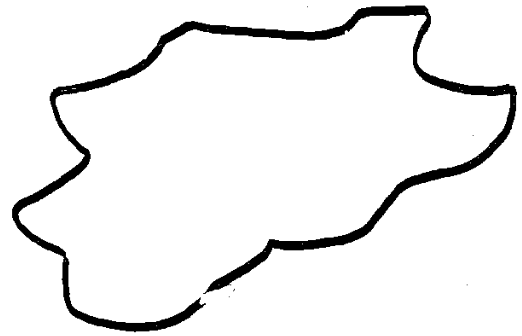
IFF Z is A UNION OF

EQUIVALENCE CLASSES

OF $\overset{*}{\equiv}_{I_{\mathcal{W}}}$.

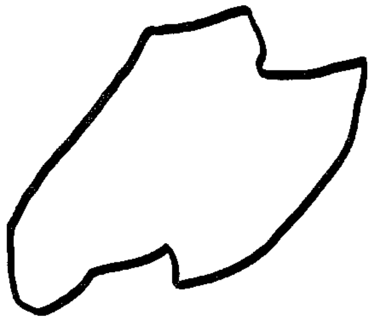
An equivalence class is called a **trace**

4'



This Z is $I_{\mathcal{W}}$-CONSISTENT

4"



This Z is **NOT** $I_{\mathcal{N}}$-CONSISTENT

⑤

THEOREM

$(\forall \mathcal{N})_{EN}$ [**FS**($\mathcal{N}$) is $I_{\mathcal{N}}$-CONSISTENT

IF ϱ OBSERVABLE IN $\mathcal{N}$,
THEN EACH ELEMENT OF
$[\varrho]_{I_{\mathcal{N}}}$ OBSERVABLE IN $\mathcal{N}$.

Each equivalence class $[\alpha]_{I_{\mathcal{N}}}$
is either ***included*** *in* **FS**($\mathcal{N}$)
or ***disjoint*** *with* $\overline{\textbf{FS}}(\mathcal{N})$

Those that are included are called
(FIRING) TRACES

FT($\mathcal{N}$)

SEQUENTIAL OBSERVATIONS

- FIRING SEQUENCES

Linear – difficult to interpret

break them down to

- DEPENDENCE GRAPHS

acyclic directed graphs

$\rightsquigarrow$

- PARTIAL ORDERS

(7) $\mathcal{N}$ $\quad \varrho \in \mathbf{FS}(\mathcal{N})$

THE CANONICAL DEPENDENCE GRAPH OF ϱ $\quad \langle \varrho \rangle_{D_{\mathcal{N}}}$

(i) $\varrho = \lambda \rightsquigarrow \langle \varrho \rangle_{D_{\mathcal{N}}}$ is empty

(ii) $\varrho = e_1 \dots e_n$, $n \geq 1$, $e_1, \dots, e_n \in E_{\mathcal{N}}$ $\rightsquigarrow$

$\langle \varrho \rangle_{D_{\mathcal{N}}}$ is the $E_{\mathcal{N}}$-lab. graph (V, Y, φ)

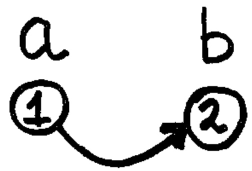
- $V = \{1, \dots, n\}$,
- $(\forall i \in \{1, \dots, n\}) [\varphi(i) = e_i]$
- $(\forall i, j \in \{1, \dots, n\})$
  $[(i,j) \in Y$ iff
  $(i<j)$ & $(e_i, e_j) \in D_{\mathcal{N}}]$

25)

$\varrho = a\ b\ c\ a\ d$

26)

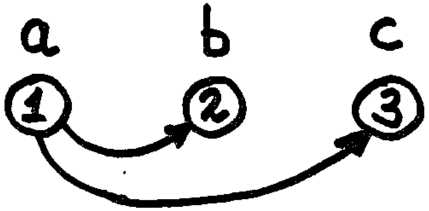
$\varrho = a\ b\ c\ a\ d$



$(a, b) \in D$

27)

$\varrho = a\ b\ c\ a\ d$

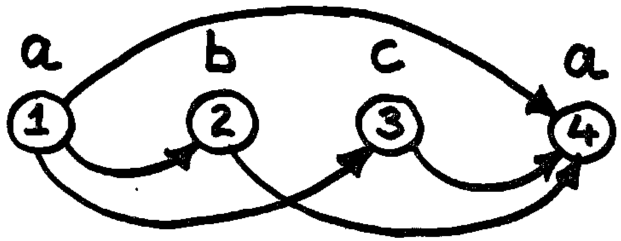


$(c, a) \in D$ $(c, b) \in I$

$(a, b) \in D$

28)

$\varrho = a\ b\ c\ a\ d$

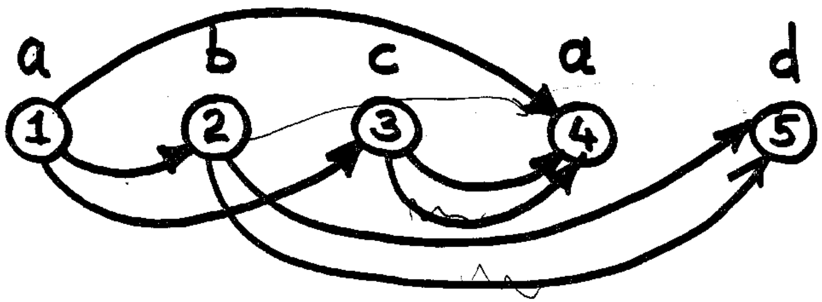


$(a, a) \in D$

$(c, a) \in D$ $(c, b) \in I$

$(a, b) \in D$

$\varrho = a\ b\ c\ a\ d$

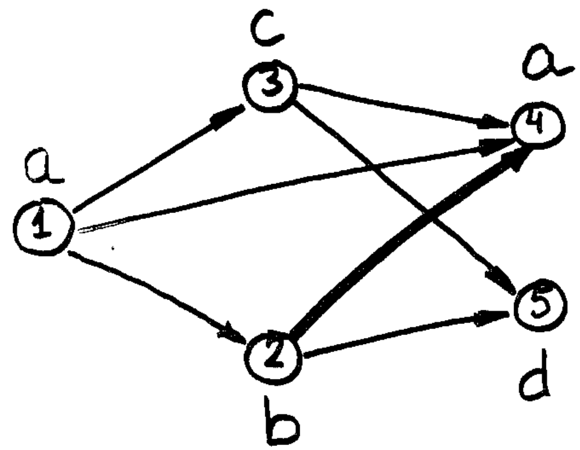


$(d,b) \in D \quad (d,c) \in D \quad (d,a) \in I$

$(a,a) \in D$

$(c,a) \in D \quad (c,b) \in I$

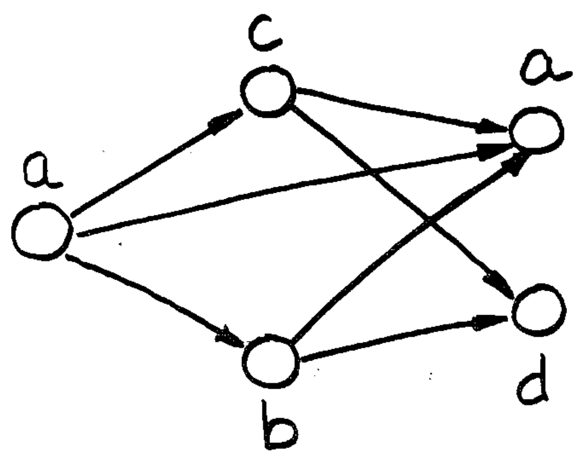
$(a,b) \in D$



$\langle \varrho \rangle_D$ the canonical dependence graph of ϱ

$\varrho = a\ b\ c\ a\ d$

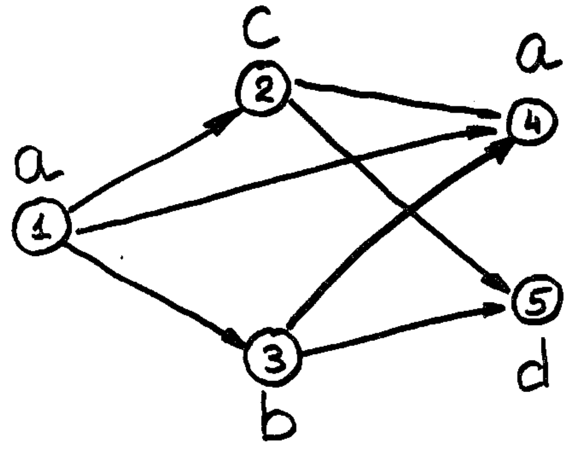
31)



$\overline{\langle \varrho \rangle}_D$

<u>abstract</u> dependence graph of ϱ

32)



$\langle \varrho' \rangle_D$

$\varrho' = a\,c\,b\,a\,d$

$\varrho = a\,\underline{b\,c}\,a\,d$ $(b,c) \in I$

$\varrho' = a\,\underline{c\,b}\,a\,d$

$\langle \varrho' \rangle_D$ <u>isom</u> $\langle \varrho \rangle_D$

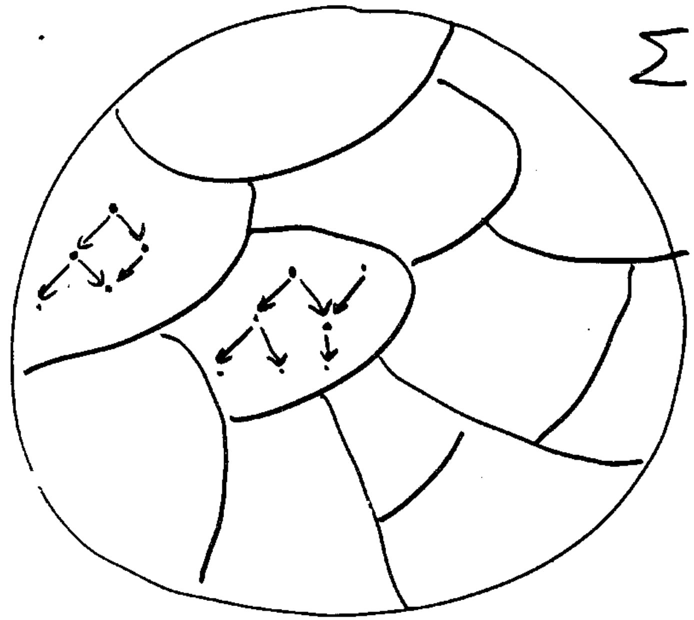
$\overline{\langle \varrho' \rangle_D} = \overline{\langle \varrho \rangle_D}$

<u>THEOREM</u>. Let $\mathcal{Z} = (\Sigma, I, D)$ and let $\varrho, \varrho' \in \Sigma^*$.

$\varrho \overset{*}{\equiv}_I \varrho'$ iff $\overline{\langle \varrho \rangle_D} = \overline{\langle \varrho' \rangle_D}$

- - - - - - -

$[\varrho]_I = [\varrho']_I$ iff $\overline{\langle \varrho \rangle_D} = \overline{\langle \varrho' \rangle_D}$



$\Sigma^* / \overset{*}{\equiv}_I$

$t \in \Theta(I) \rightsquigarrow \overline{\langle t \rangle_D}$

abstract dependence graph of t

<u>adg</u>(t)

35)

POSETS

(A, R) ANTISYM. TRANSIT. REFLEX.

$g = (V, E)$ DIR. ACYCL. GRAPH

TRANS. & REFL. CLOS.

$\leqslant_g = (V, E^*)$

36)

$\varrho \in \Sigma^*$

$\langle \varrho \rangle_D$ $\langle\langle \varrho \rangle\rangle_D$

$\overline{\langle \varrho \rangle}_D$ $\overline{\langle\langle \varrho \rangle\rangle}_D$

$\underline{adg}(t)$ $\underline{alp}(t)$

ADG(T) **ALP**(T)

37)

$\mathbf{FS}(\mathcal{N}) \subseteq E^*$

$[\mathbf{FS}(\mathcal{N})]_{I_{\mathcal{N}}}$ — FIRING TRACES OF $\mathcal{N}$

$\mathbf{FT}(\mathcal{N})$

$\mathbf{ADG}(\mathbf{FT}(\mathcal{N}))$ — ABSTRACT FIRING DEP. GRAPHS OF $\mathcal{N}$

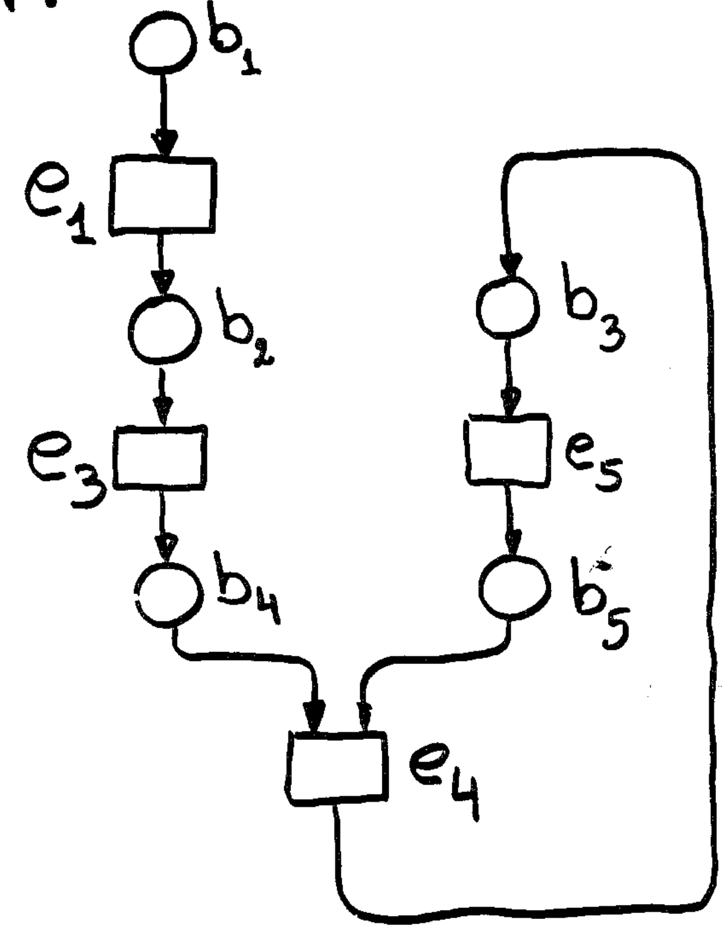
$\mathbf{AFD}(\mathcal{N})$

$\mathbf{ALP}(\mathbf{FT}(\mathcal{N}))$ — ABSTRACT FIRING LAB. POSETS OF $\mathcal{N}$

$\mathbf{AFLP}(\mathcal{N})$

38)

$\mathcal{N}$:



$C_{in} = \{b_1, b_3\}$

39)

$$I_{\mathcal{N}} = \{(e_1, e_5), (e_5, e_1), (e_1, e_4), (e_4, e_1), (e_3, e_5), (e_5, e_3)\}$$

$$D_{\mathcal{N}} = E \times E - I_{\mathcal{N}}$$

40)

$$\varrho = e_1 e_5 e_3 e_4 e_5 \in \mathbf{FS}(\mathcal{N})$$

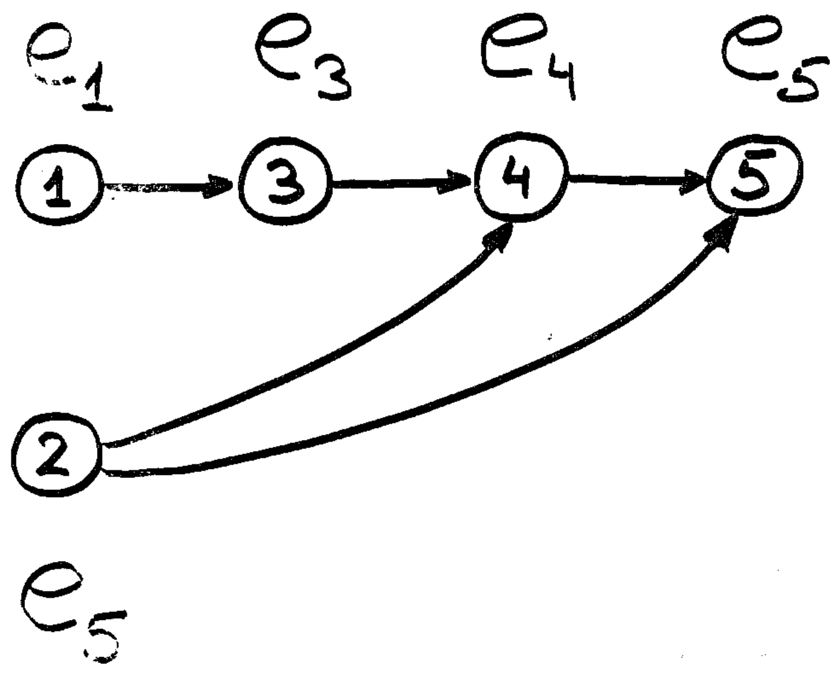
$$[\varrho]_{I_{\mathcal{N}}} = \{e_1 e_5 e_3 e_4 e_5, \; e_5 e_1 e_3 e_4 e_5, \; e_1 e_3 e_5 e_4 e_5\}$$

41)

$$[\varrho]_{I_{\mathcal{N}}} \subseteq \mathbf{FS}(\mathcal{N})$$

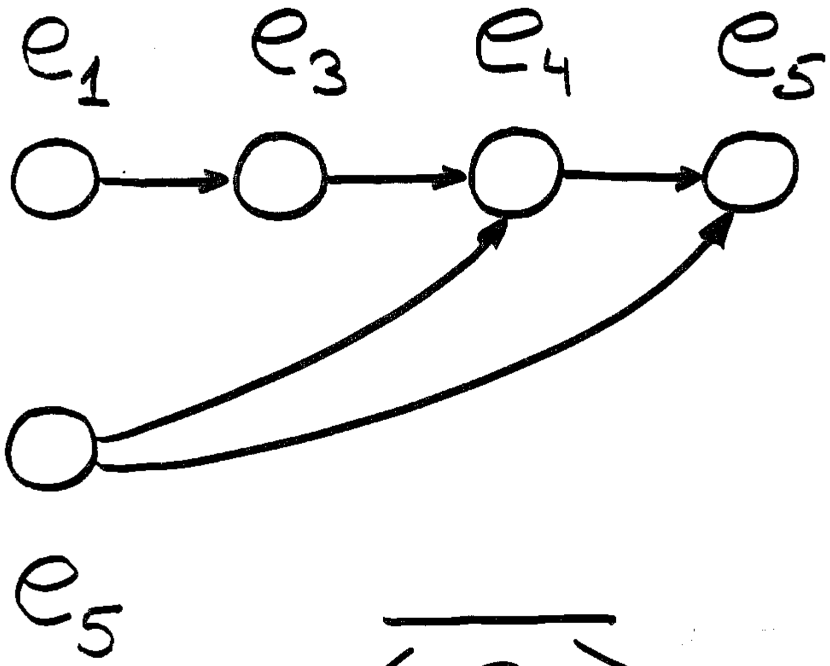
$$[\varrho]_{I_{\mathcal{N}}} \in \mathbf{FT}(\mathcal{N})$$

42)



$$\langle \varrho \rangle_{D_{\mathcal{N}}}$$

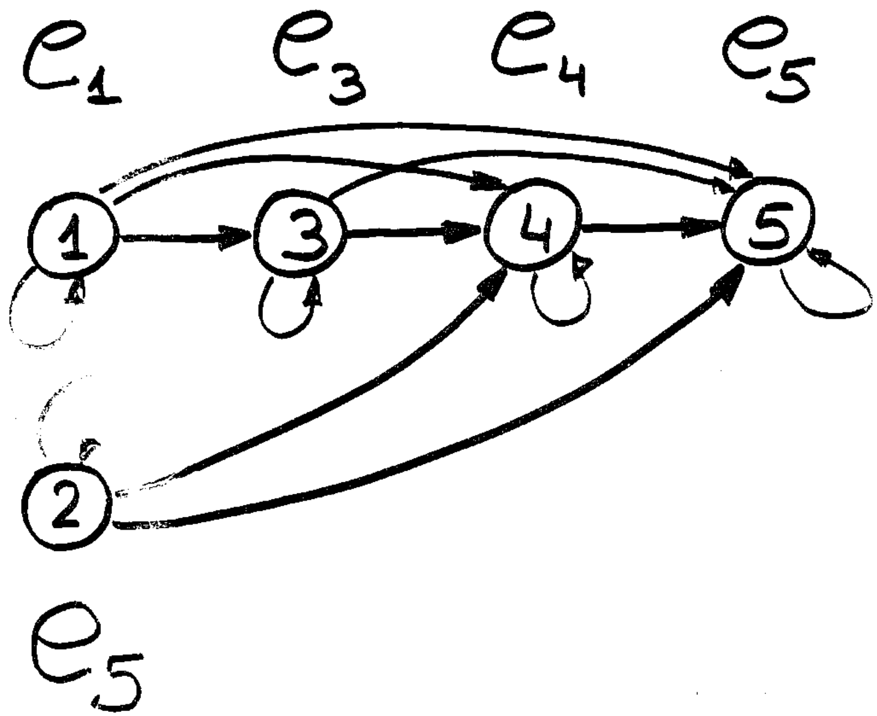
43)



$$\overline{\langle e \rangle}_{\mathcal{D}_{\mathcal{N}}}$$

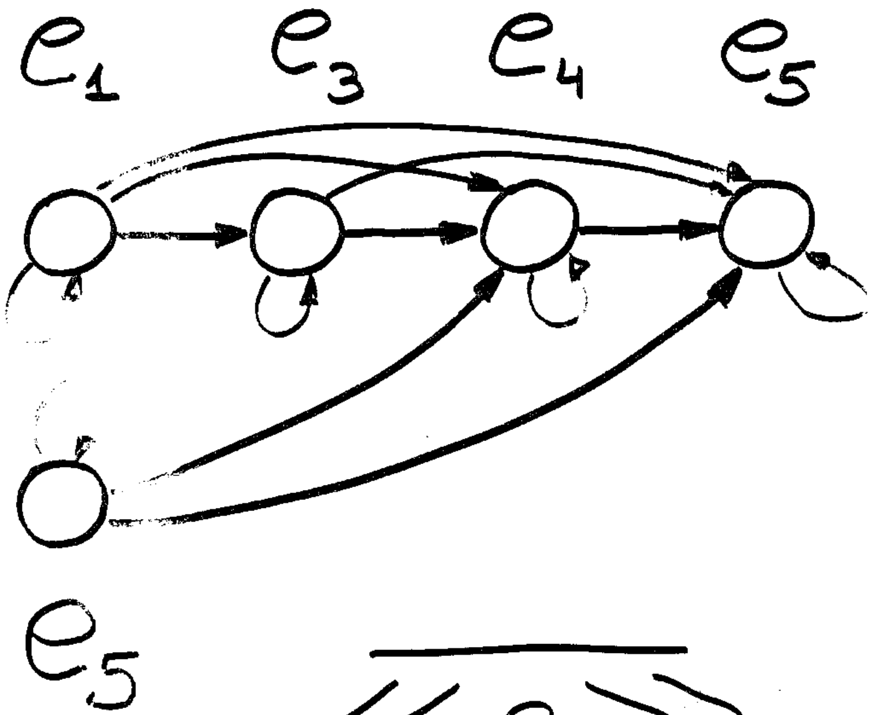
$$\in \mathbf{AFD}(\mathcal{N})$$

44)



$$\langle\langle e \rangle\rangle_{\mathcal{D}_{\mathcal{N}}}$$

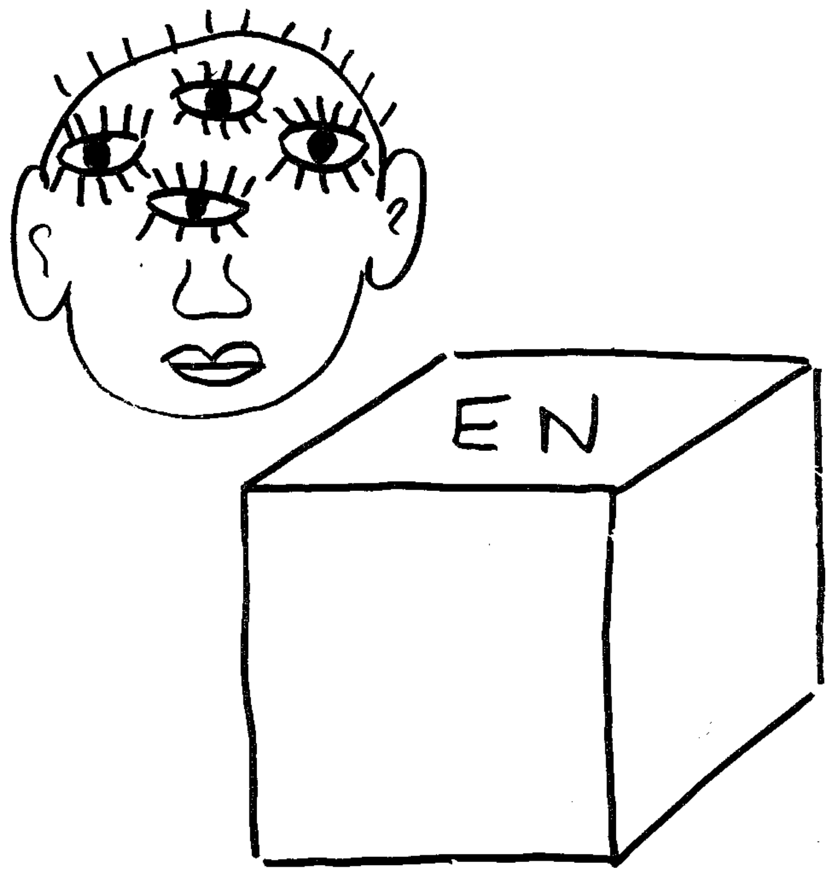
45)



$$\overline{\langle\langle e \rangle\rangle}_{D_{\mathcal{N}}}$$

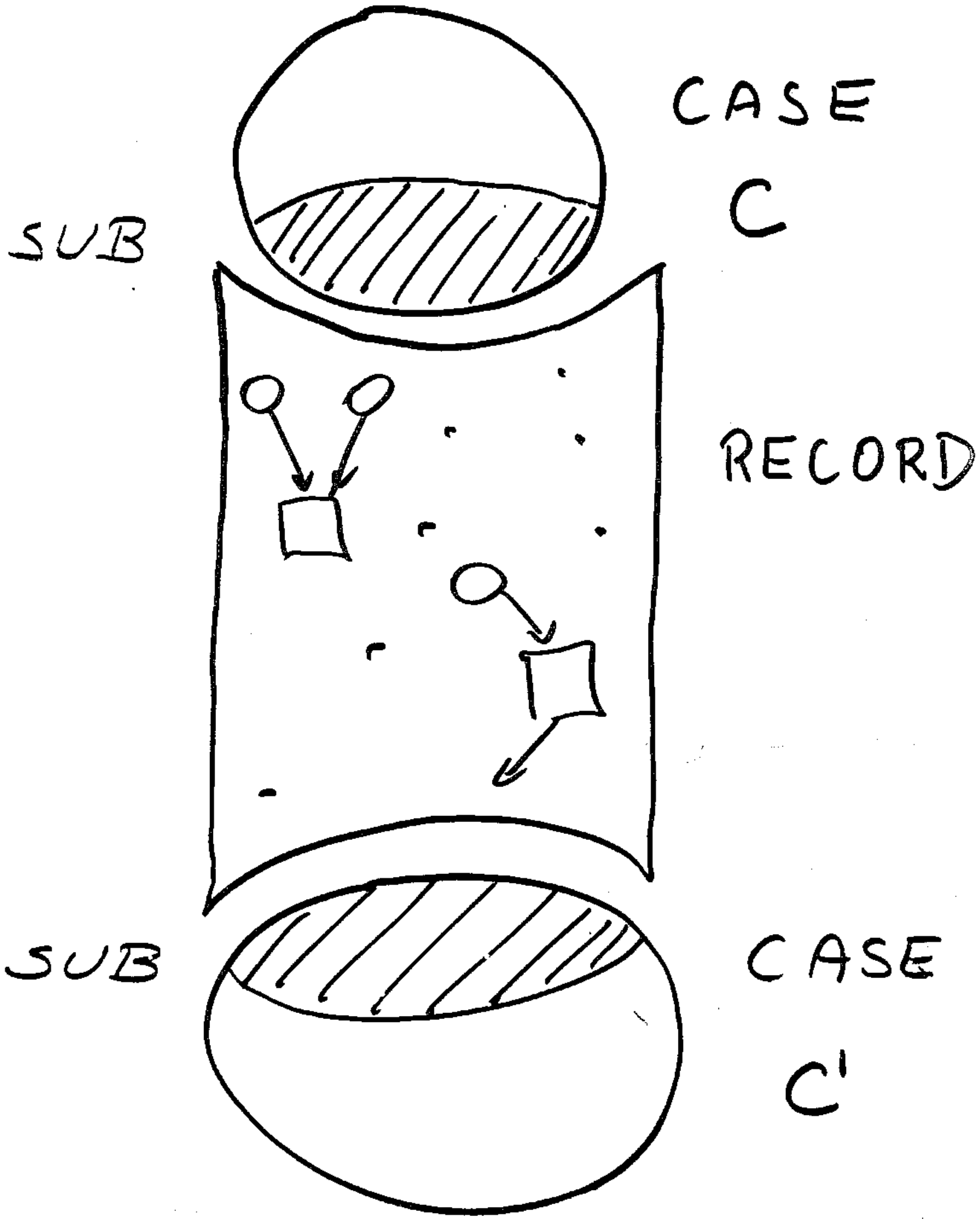
$$\in \mathbf{AFLP}(\mathcal{N})$$

46)

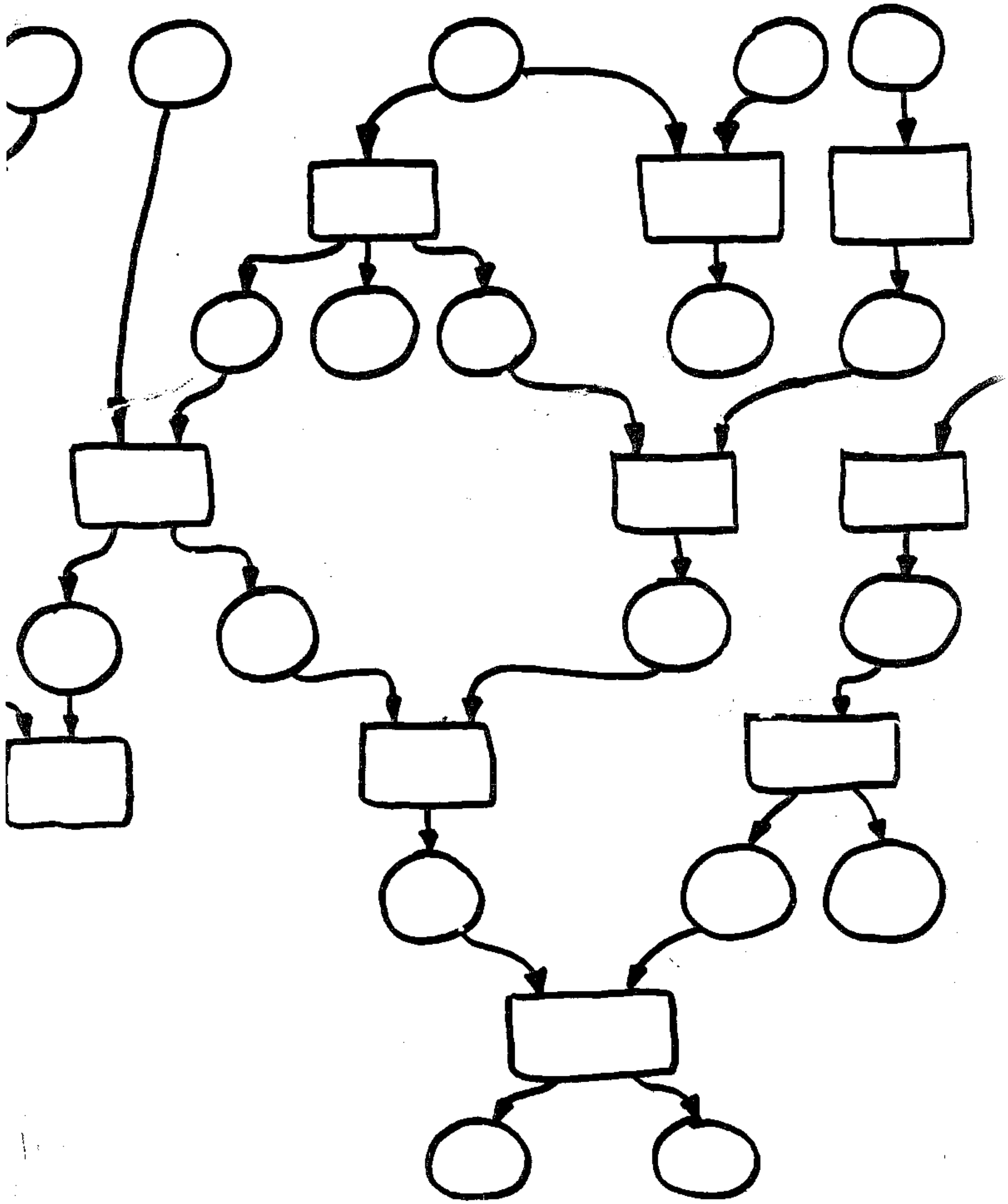


NON-SEQUENTIAL OBSERVATIONS

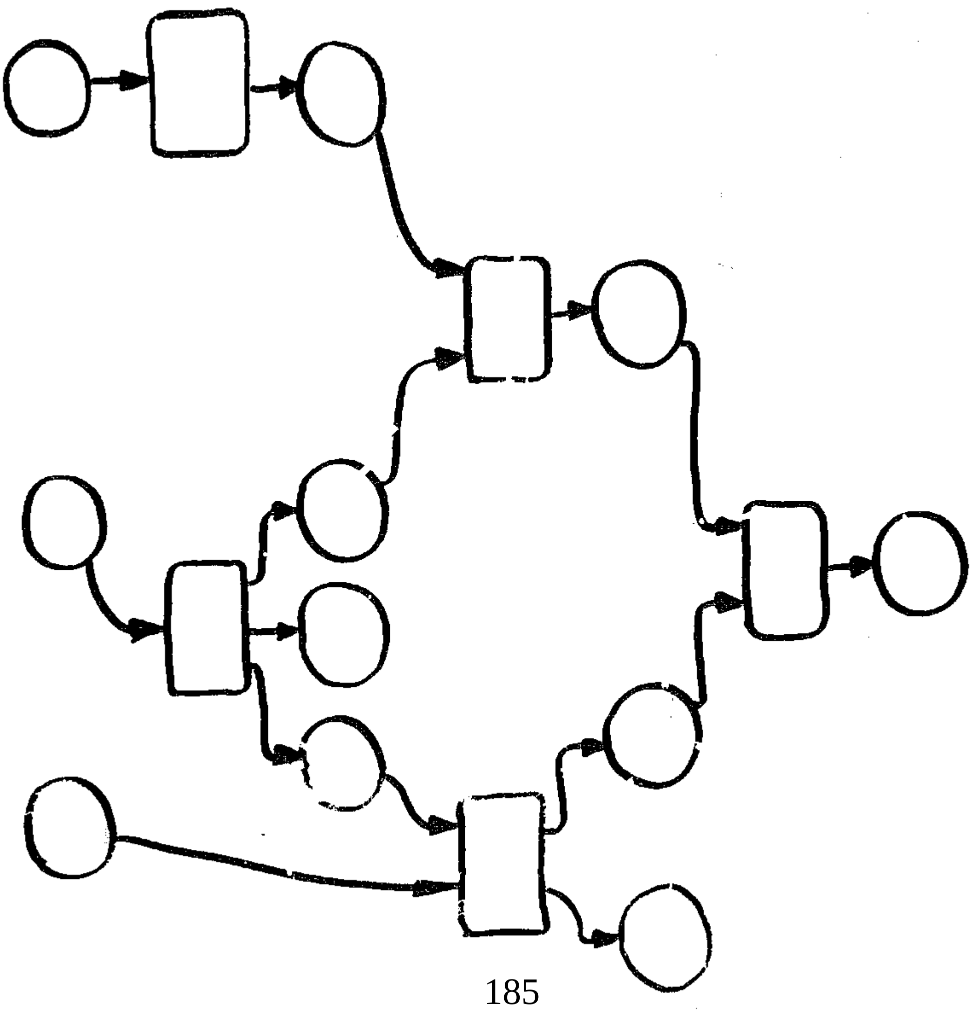
47)



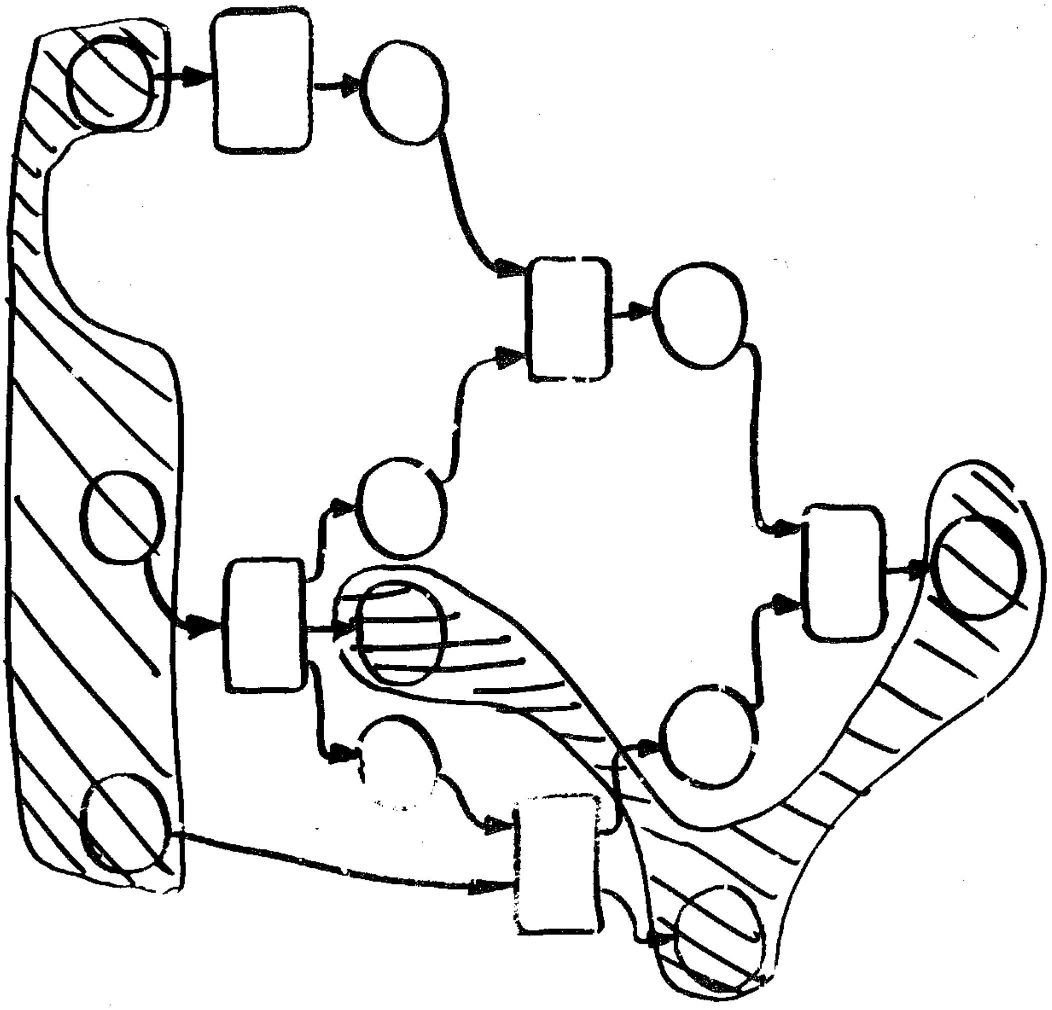
48)



49)



50)



51.

A NET $N=(S,T,F)$
IS AN OCCURRENCE
NET IFF

$(\forall s \in S)[\,|{}^{\bullet}s| \leq 1$ AND
$|s^{\bullet}| \leq 1\,]$

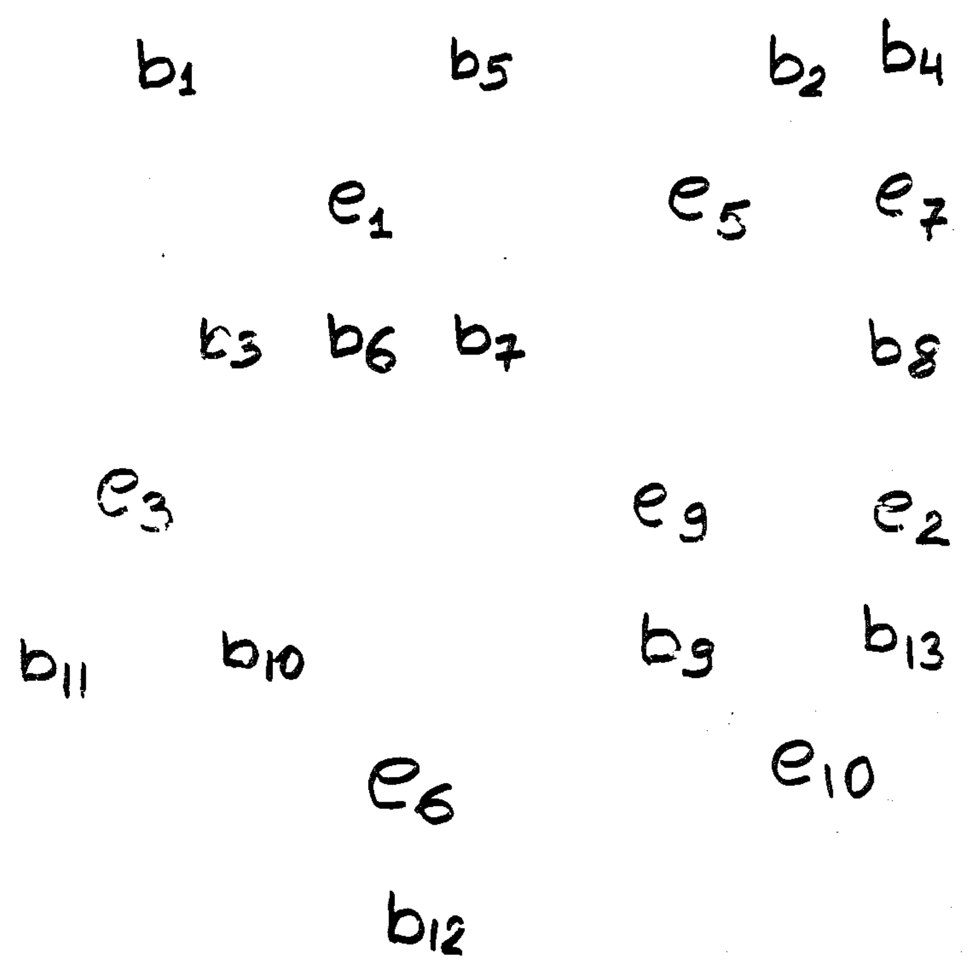
S-NON-BRANCHING

$(\forall x,y \in X)[(x,y) \in F^{+} \Longrightarrow$
$(y,x) \notin F^{+}]$

ACYCLIC

$(\forall t \in T)[\,t^{\bullet} \neq \emptyset\,]$

52)



53.

b_1 b_5 b_4

b_3 b_7 b

b_6

b_{11} b_{10} b_9

b_{12}

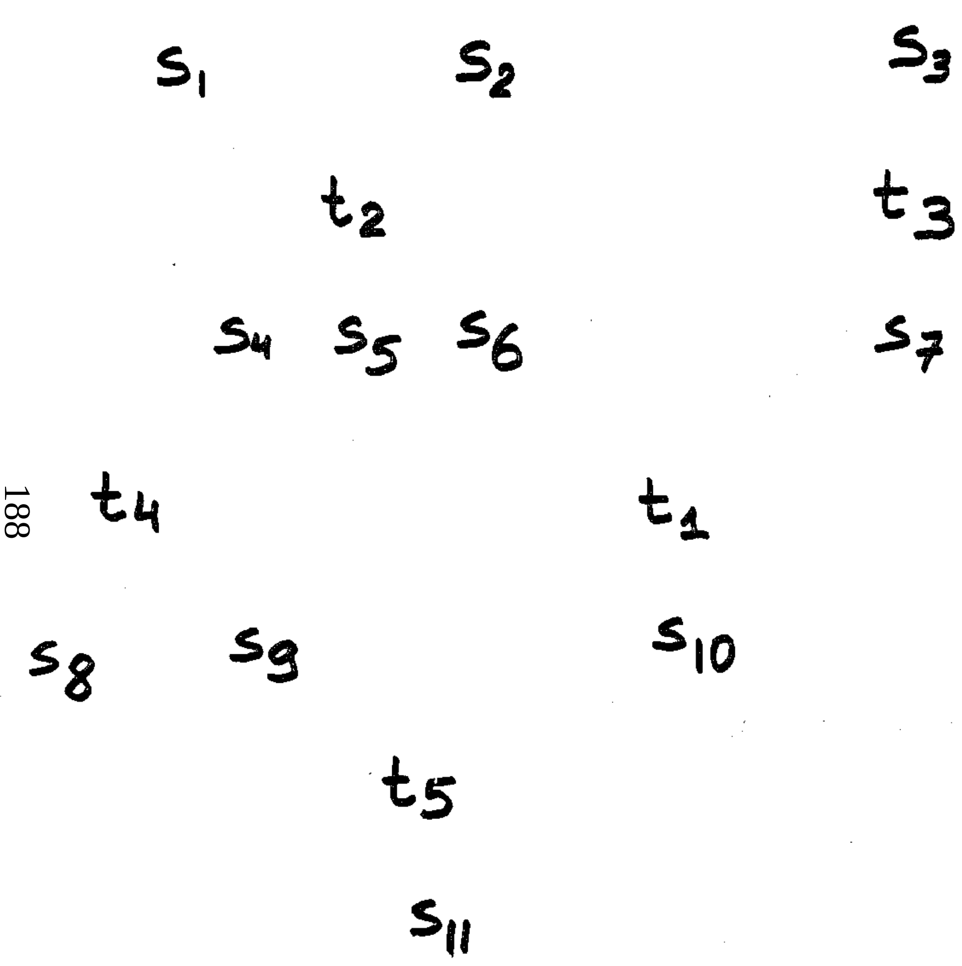
54)

e_1 e_7

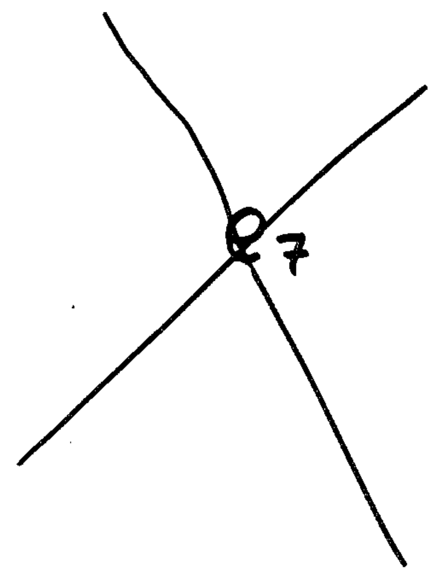
e_3 e_9

e_6

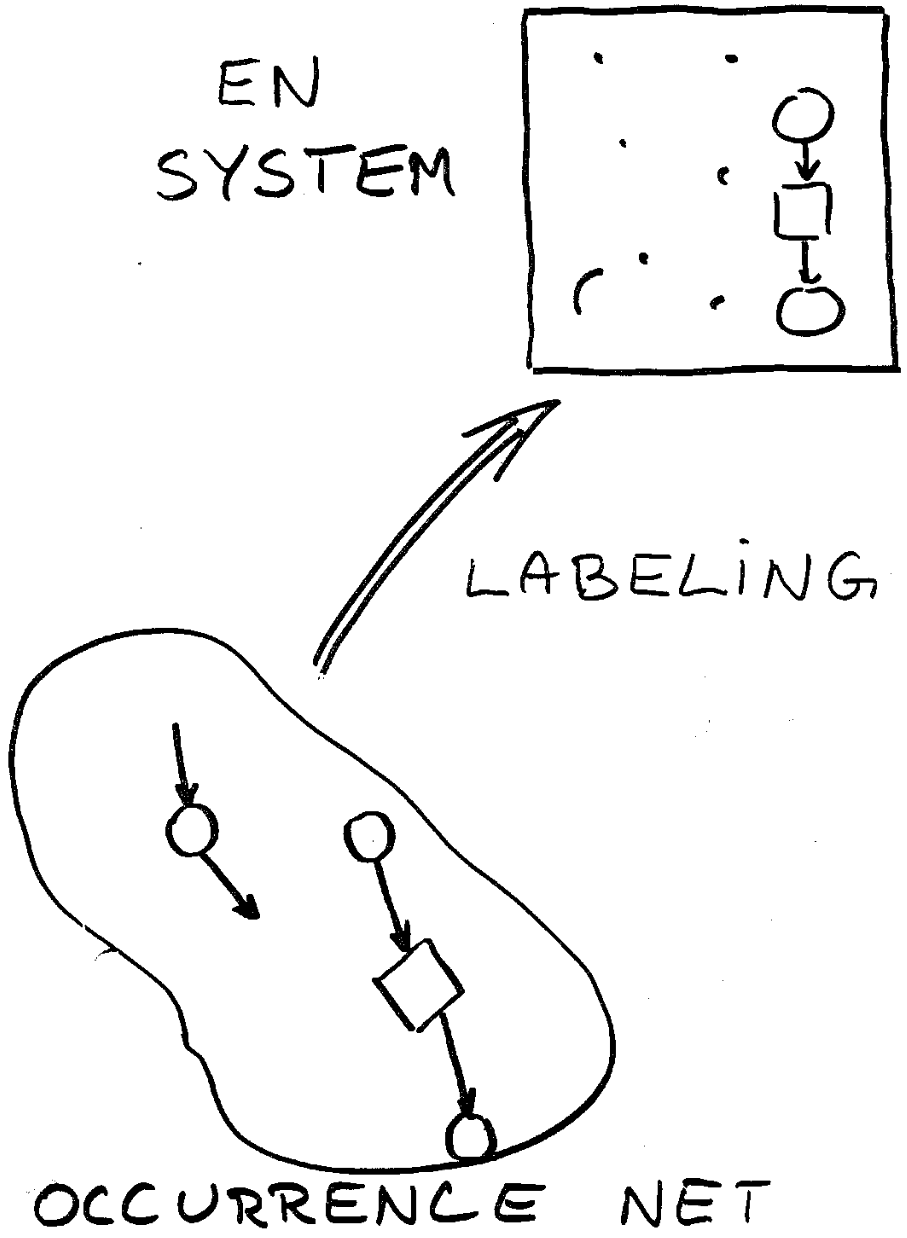
55)



56)



57)



58.

A NODE-LABELED OCCURRENCE NET

$N = (S, T, F, \varphi)$

(S, T, F) OCCURR. NET

$\varphi : S \cup T \rightarrow \Sigma$

$\varphi(S) \cap \varphi(T) = \emptyset$

59)

$\mathcal{N}$ EN SYSTEM

$N = (S, T, H, \varphi)$ NODE-LAB. OCCUR. NET

N is A PROCESS OF $\mathcal{N}$ IF

(i) $\varphi(S) \subseteq B_{\mathcal{N}}$ AND $\varphi(T) \subseteq E_{\mathcal{N}}$.

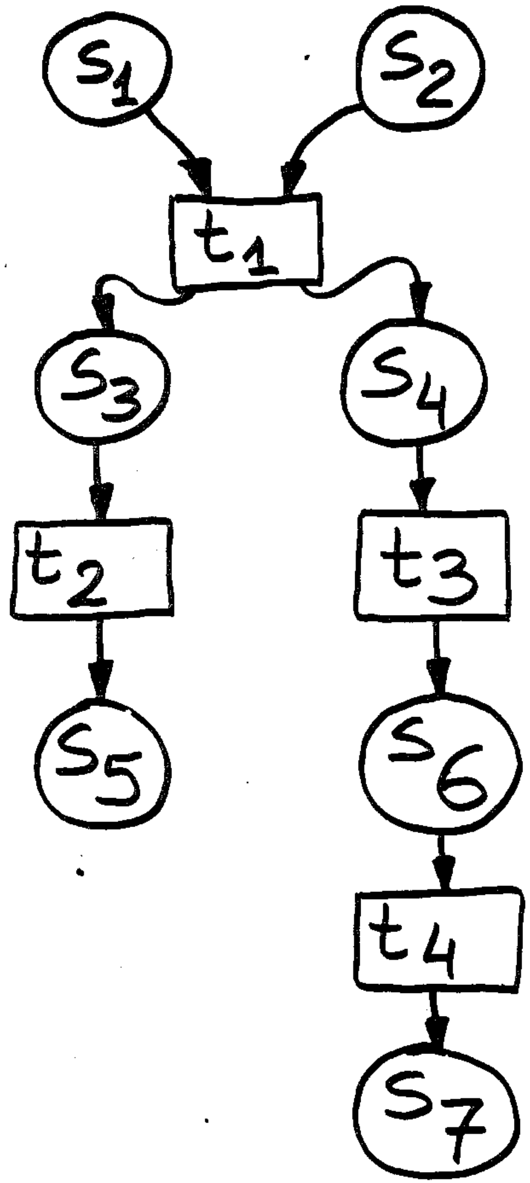
(ii) $(\forall s_1, s_2 \in S)\ [\varphi(s_1) = \varphi(s_2)$
$\implies (s_1 \leq_N s_2) \vee (s_2 \leq_N s_1)]$

(iii) $(\forall t \in T)\ [\varphi({}^{\bullet}t) = {}^{\bullet}\varphi(t)$
AND $\varphi(t^{\bullet}) = \varphi(t)^{\bullet}\ ]$.

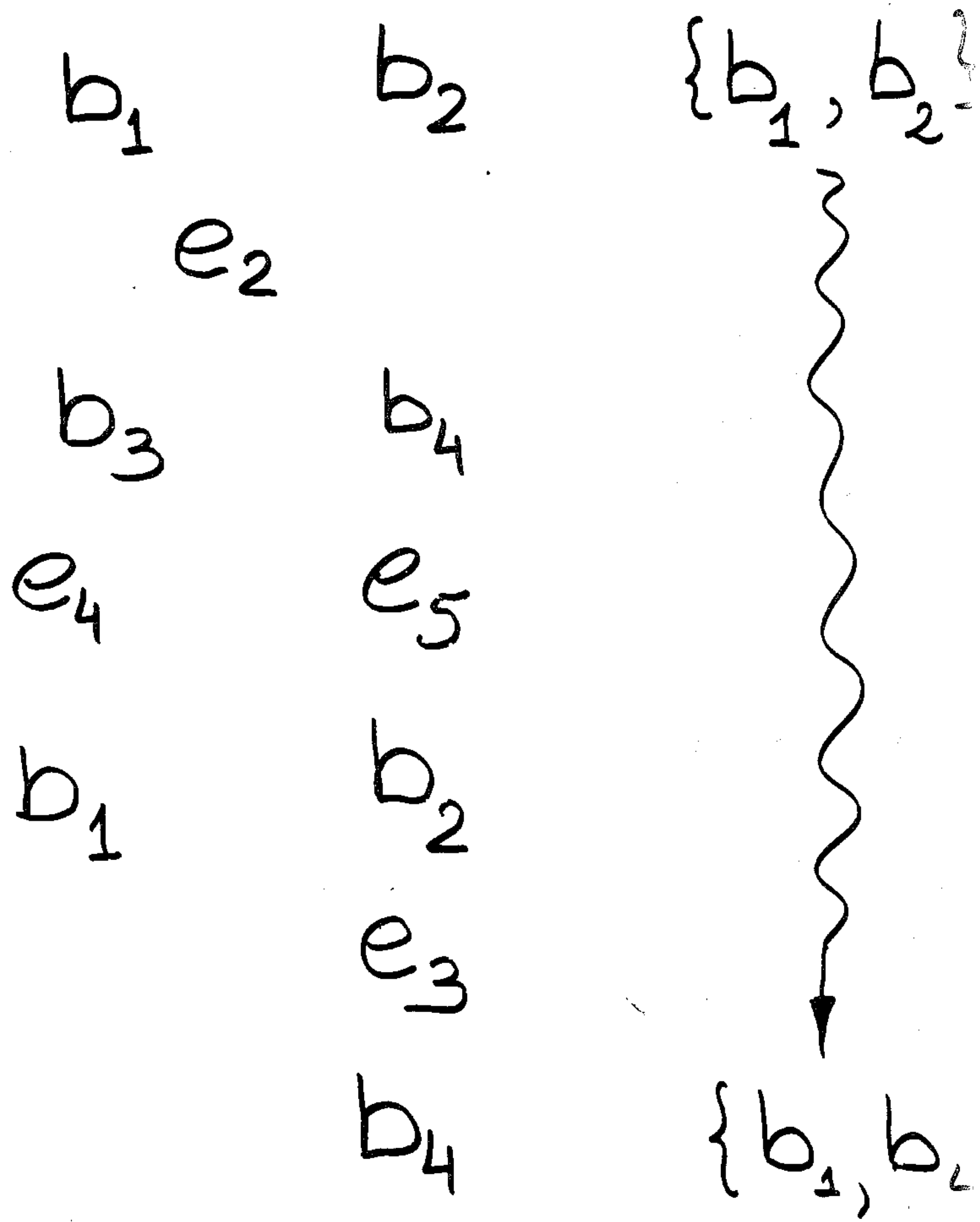
(iv) $\varphi({}^{0}N) \subseteq C_{in}$.

$\mathbf{P}(\mathcal{N})$

60)



61)



62)

<u>THEOREM</u>

$\mathcal{N}$ EN SYSTEM

$N = (S, T, F, \varphi) \in \mathbf{P}(\mathcal{N})$

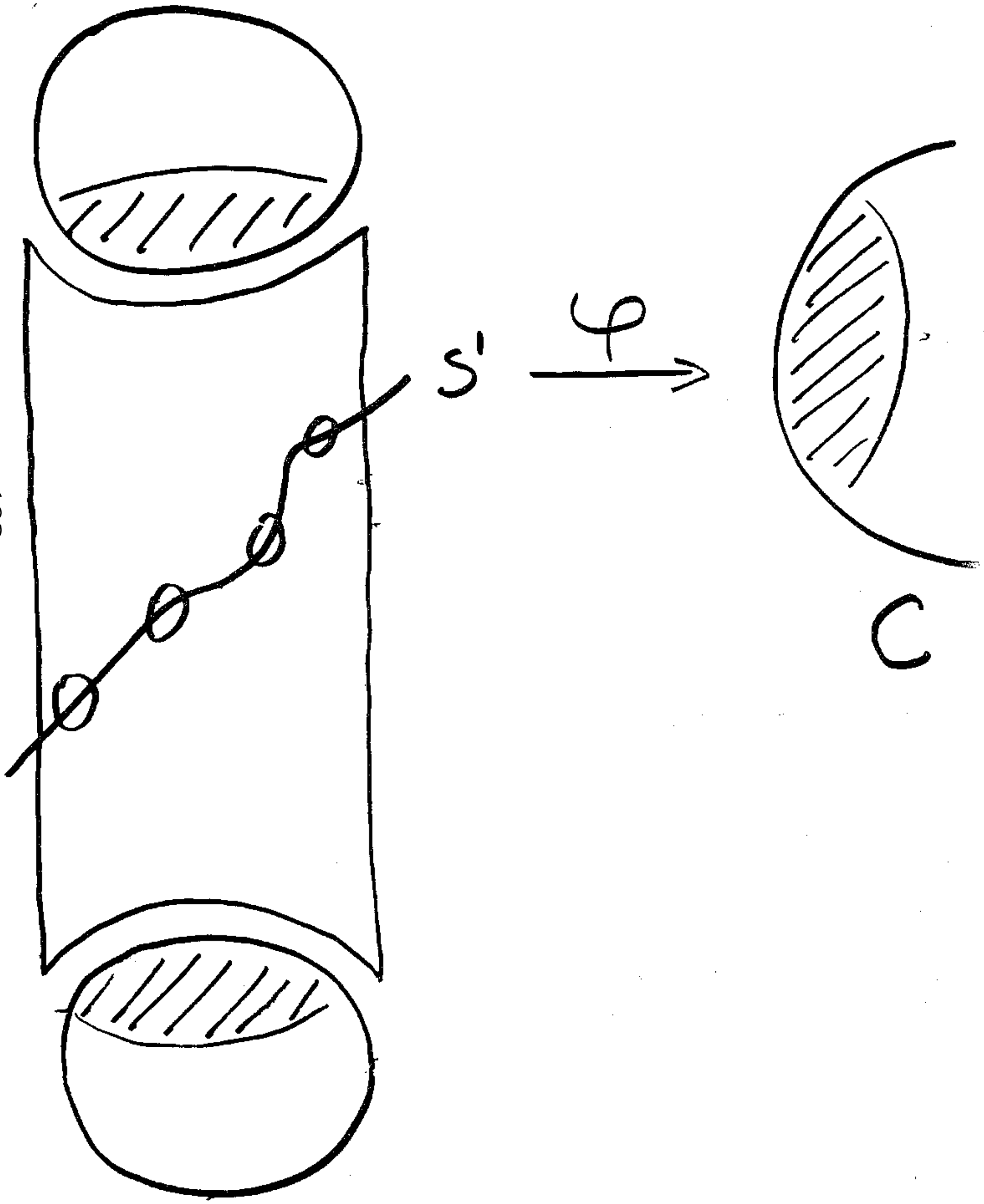
$S' \subseteq S$ A SLICE OF N

—

$(\exists C \in \mathcal{C}_{\mathcal{N}})$

$[\varphi(S') \subseteq C]$.

63)



64)

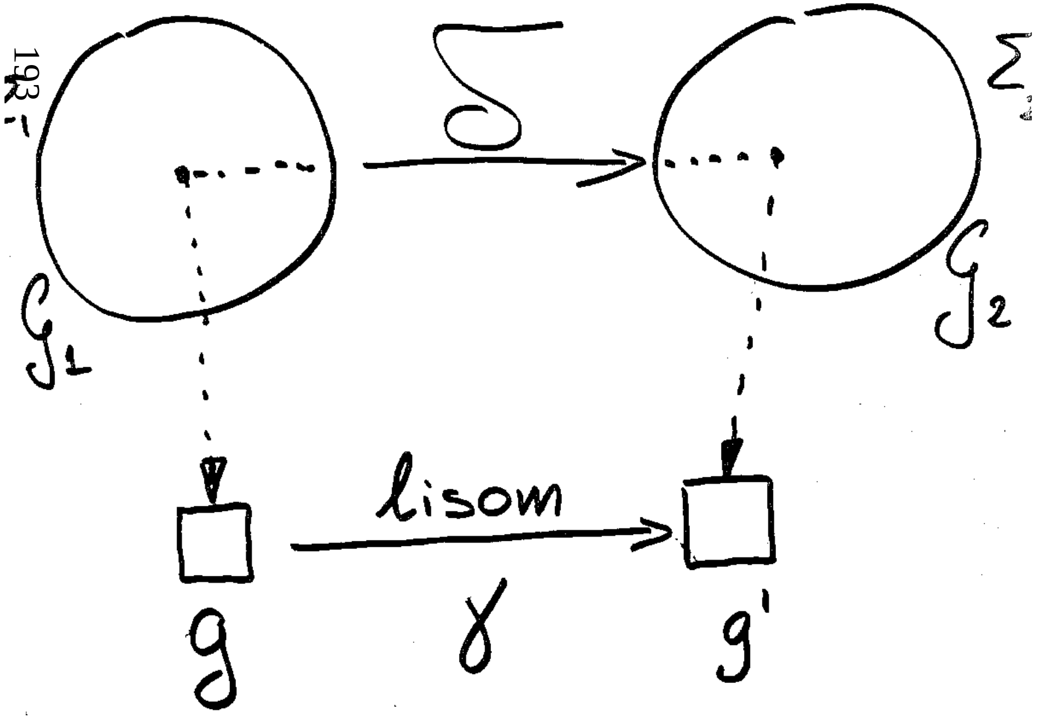
EN SYSTEM $\mathcal{N}$ is REDUCED IFF ALL EVENTS OF $\mathcal{N}$ "VISIBLE" IN $SCG(\mathcal{N})$

$$\left(E_{\mathcal{N}} = \bigcup_{u \in \mathcal{U}_{\mathcal{N}}} u \, . \right)$$

$\mathcal{G}_1$ <u>LISOM</u> $\mathcal{G}_2$

$\exists\, \gamma : \Sigma_1 \longrightarrow \Sigma_2$

$\exists\, \delta : \mathcal{G}_1 \longrightarrow \mathcal{G}_2$



EN SYSTEMS $\mathcal{N}_1, \mathcal{N}_2$
STRUCTURALLY
SIMILAR

$$\mathcal{N}_1 \equiv \mathcal{N}_2$$

<u>und</u>($\mathcal{N}_1$) <u>isom</u> <u>und</u>($\mathcal{N}_2$)

φ

C^1_{in}, C^2_{in} RELATED
ACCORDINGLY

67)

THEOREM

$\mathcal{N}_1, \mathcal{N}_2$ REDUCED EN S's

$\mathbf{P}(\mathcal{N}_1)$ LISOM $\mathbf{P}(\mathcal{N}_2)$

IFF

$\mathcal{N}_1 \equiv \mathcal{N}_2$ ∎

PROCESS REPRES. OF THE BEHAVIOUR OF AN EN SYSTEM IS TOO DETAILED

68)

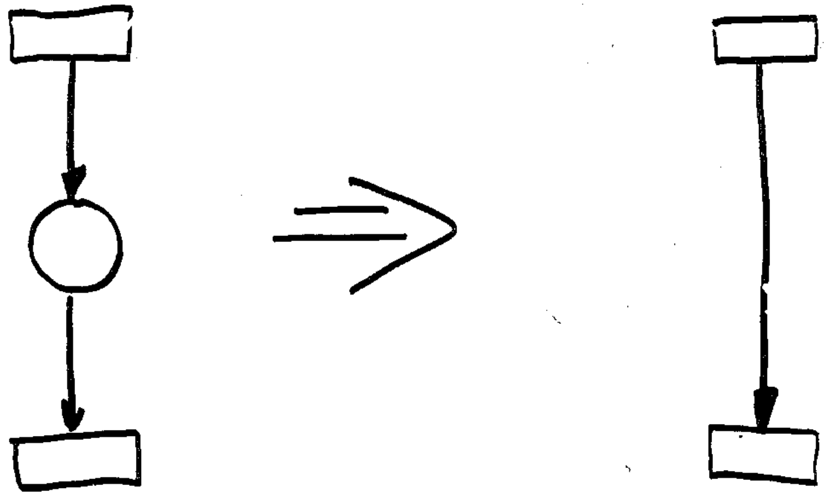
g A BIPARTITE GRAPH

$g = (V, W, F)$

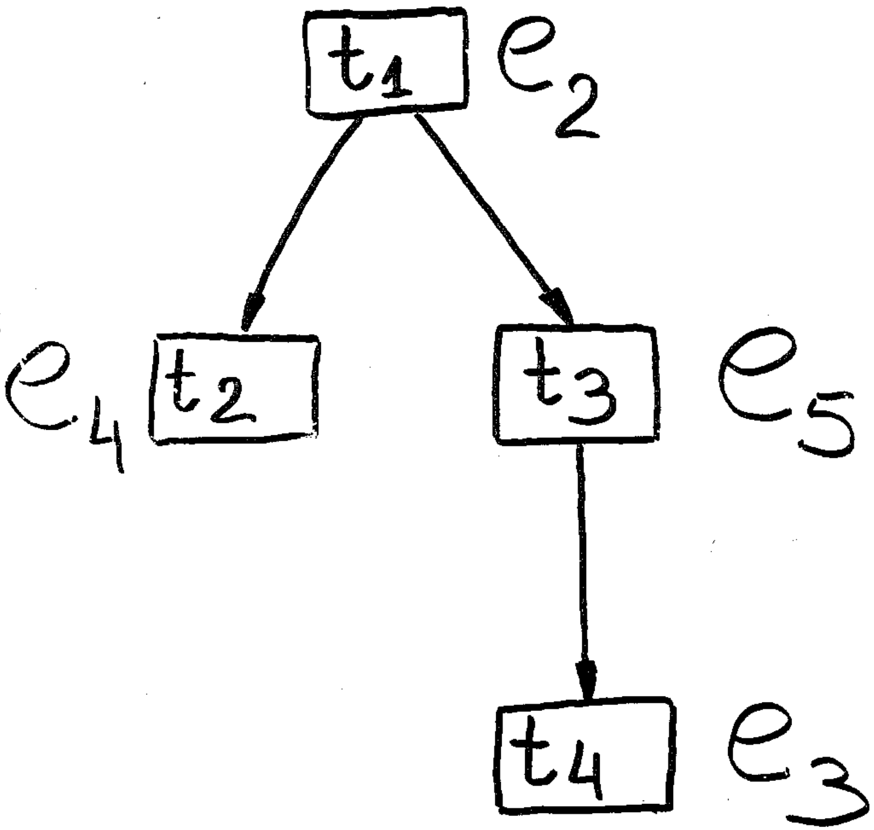
W-CONTRACTION OF g

$\underline{ctr}_W(g) = (V, F')$

F: F':



69)



70)

$\mathcal{N}$ AN EN SYSTEM

$N \in \mathbf{P}(\mathcal{N})$ $S = S_N$

THE S-CONTRACTED VERSION OF N IS A CONTRACTED PROCESS OF $\mathcal{N}$

CP$(\mathcal{N})$

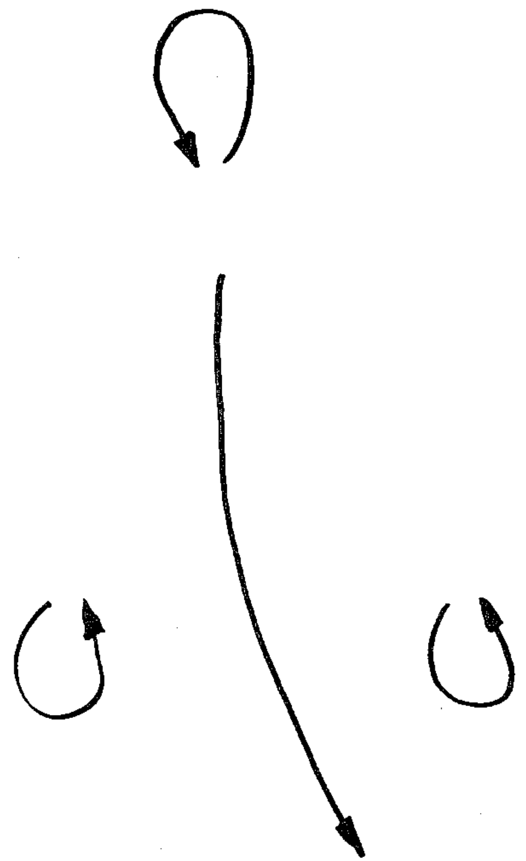
THE LABELED POSET

$\leq_{\underline{ctr}_S(N)}$ IS AN

ELEMENTARY EVENT STRUCTURE OF $\mathcal{N}$

EES$(\mathcal{N})$

71)



72)

THEOREM

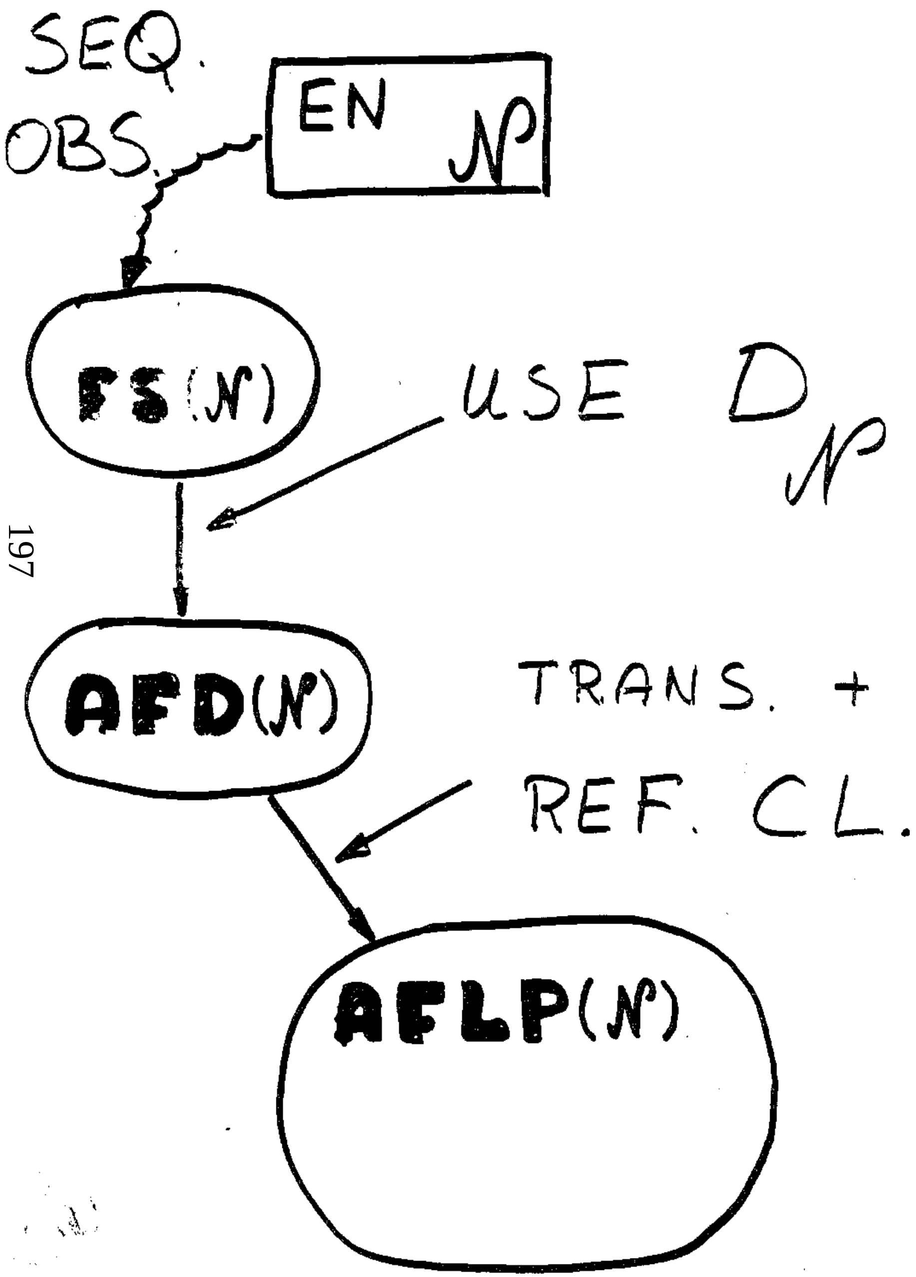
$(\exists \mathcal{N}_1, \mathcal{N}_2)_{\text{EN REDUCED}}$

$[\ \mathbf{CP}(\mathcal{N}_1)$ <u>LISOM</u> $\mathbf{CP}(\mathcal{N}_2)$

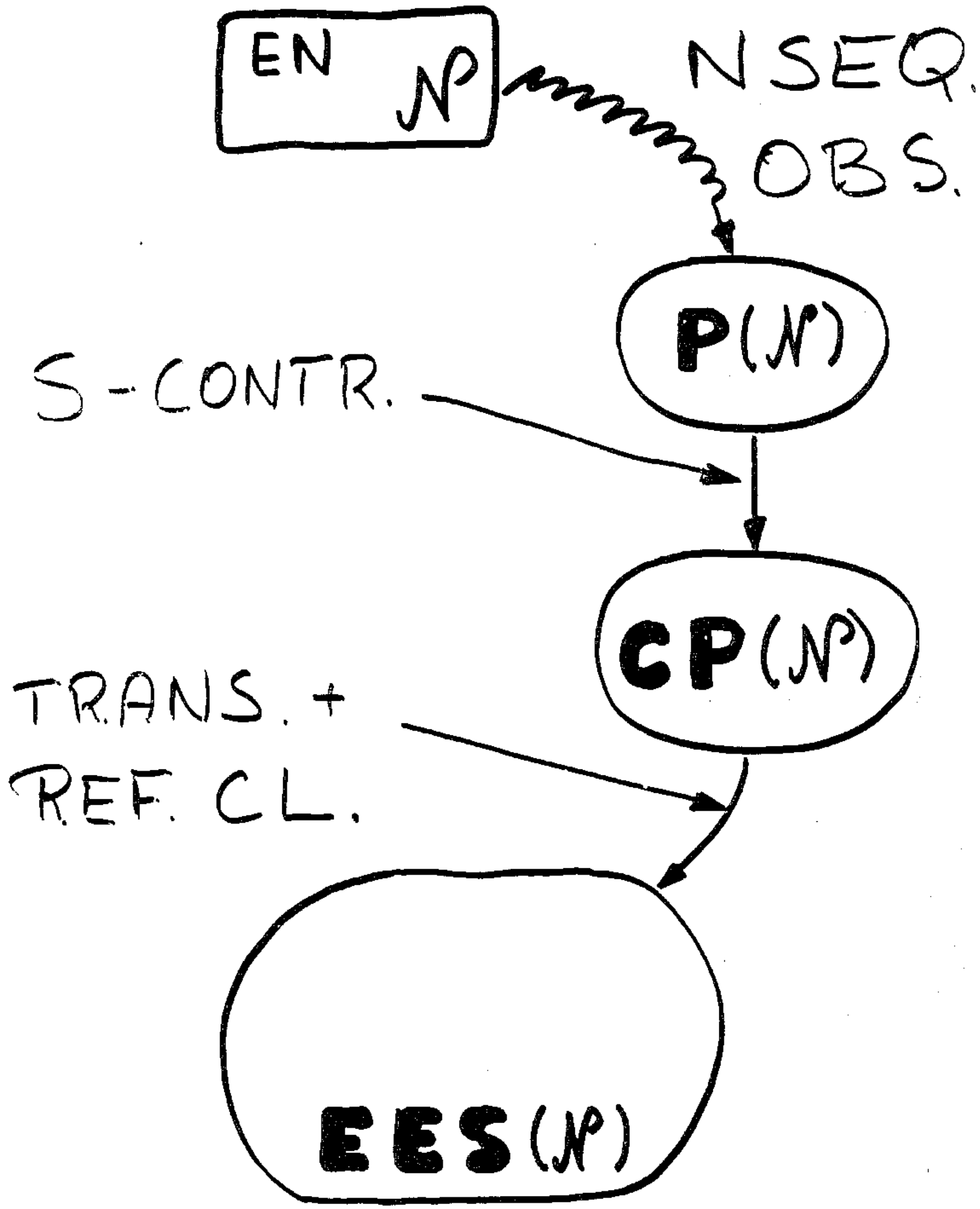
AND

$\mathcal{N}_1 \not\equiv \mathcal{N}_2\ ]$

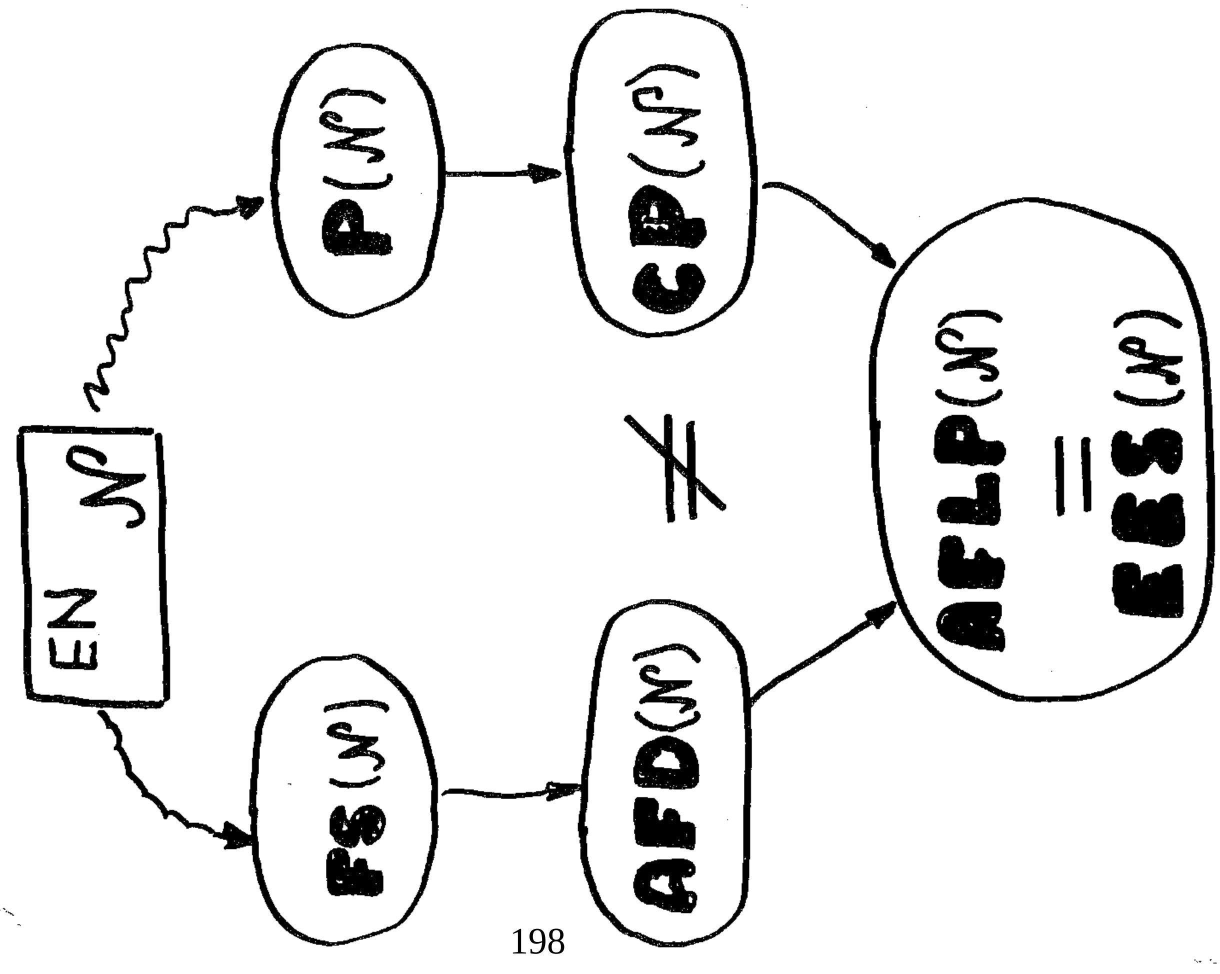
73)


SEQ.
OBS.
EN 𝒩
FS(𝒩)
USE D𝒩
AFD(𝒩)
TRANS. +
REF. CL.
AFLP(𝒩)

74)


EN 𝒩
NSEQ.
OBS.
P(𝒩)
S-CONTR.
CP(𝒩)
TRANS. +
REF. CL.
EES(𝒩)

75)



Place/Transition Nets II

M. Silva
Universidad de Zaragoza

OUTLINE

1 Introduction and motivation

1.1 Basic approaches: lead to (possibly weak) necessary *or* sufficient conditions.

1.2 On net subclasses: (sometimes) lead to necessary *and* sufficient conditions.

1.3 Nets and net systems: graph and convex geometry/linear algebra perspectives.

2 Reduction

2.1 Basic concept: Net and marking transformation. Properties preserved by a rule.

2.2 A very simple kit of reduction rules preserving liveness and boundedness.

2.3 Implicit places: Basic concept and search technique.

2.4 Reduction on net subclasses can be complete: The free choice example.

3 Structure Theory: A convex geometry/linear algebra perspective

1 Introduction and Motivation

1.1 Motivation and basic approaches

☞ State enumeration analysis:

- Reachability graph (for bounded systems)
- Coverability graph (leading to semidecision algorithms for unbounded systems)

☞ Problems:
- computationally complexs (may be intractable)
- valid only for a given initial marking.

☞ New idea: Bridge behaviour and structure

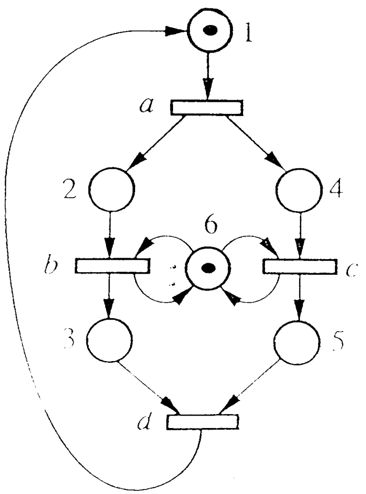
☞ The basic approaches (considered here):

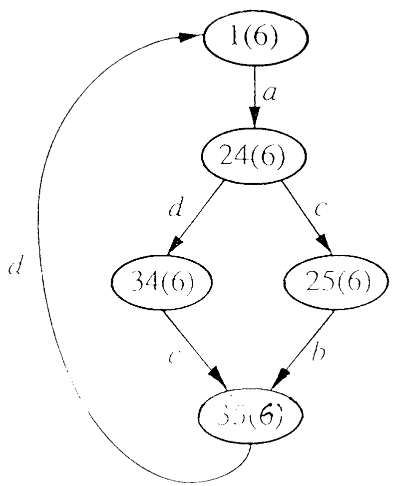
- *Reduction*: looking for "simpler" systems preserving the properties under study.
- *Global structural approaches*: Nets as graphs, nets as non-negative integer linear equations.

Reachability graph: Behaviour Sequentialization

☞ Idea: Exhaustive sequential state exploration

EXAMPLE: Adding p_6 does not change the reachability graph, but *b* and *c* cannot be concurrently fired.





☞ Problems with the new approaches:

- Reduction may be *not complete* for a given kit of reduction rules and a given net system. Therefore other analysis techniques are needed.
- Global structural analysis *can be unable to decide* on easy properties like boundedness (while decisions can be obtained for non-liveness for unbounded systems, for example).

☞ Therefore: The practical analysis of general net system may need the *cooperative* use of several analysis techniques.

☞ Nevertheless, *necessary and sufficient* conditions in the characterization of some properties are available for net *subclasses* (i.e. simpler nets).

1.2 *On net subclasses: (sometimes) lead to necessary and sufficient conditions*

☞ For "simpler" systems, the analysis problem became "easier".

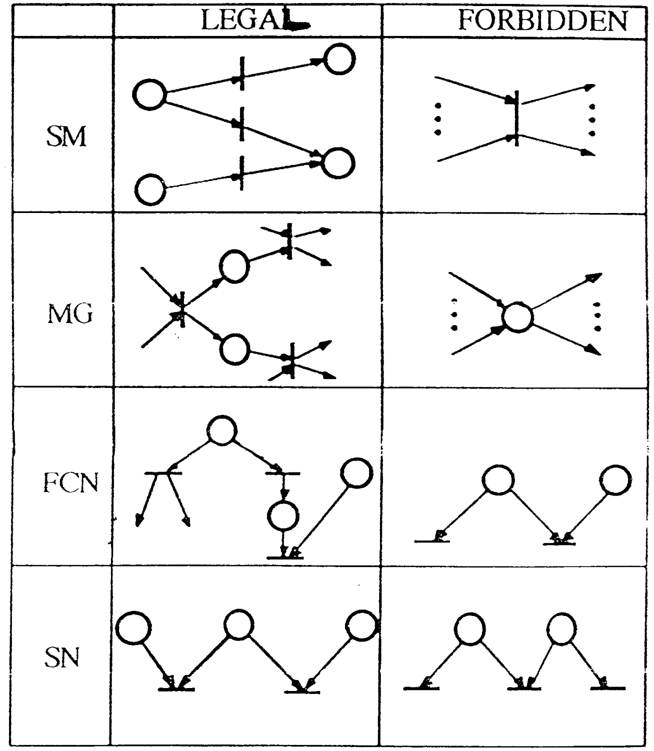
(e.g. continuous time, constant coefficient linear differential equations are easy to solve).

☞ Simpler systems?

Are defined constraining the interleaving of concurrency and conflicts.

☞ Only the most basic/classical net subclasses will be considered. They define the constraints in a syntactical way.

Net subclasses



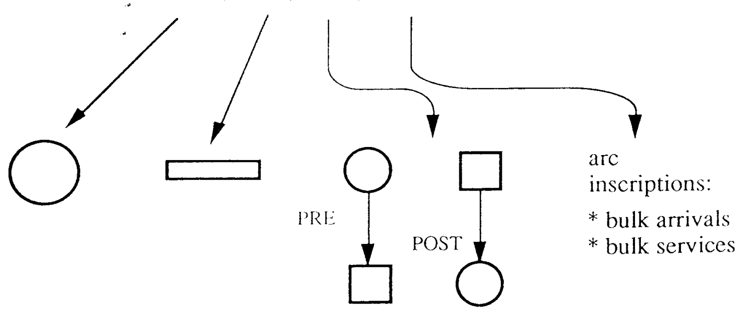
1.3 *Nets and net systems*

☞ Net system ≡ Net structure +
+ distributed initial state (marking)

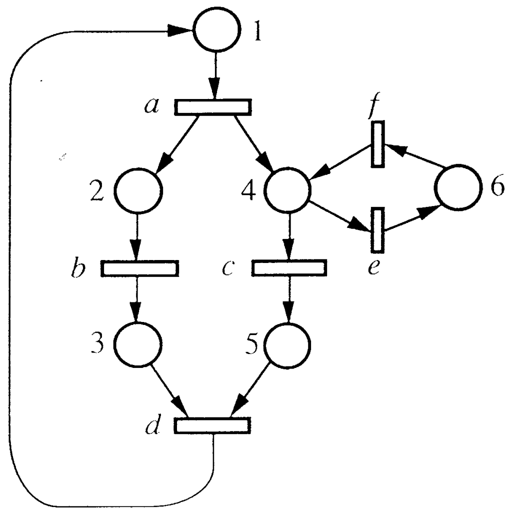
☞ Discrete-event-dynamic-system:
- * States → state variables: Places, P
- * Events → state-transitions: Transitions, T

☞ N describes a *net structure* (weighted-bipartite directed graph):

$$N = < P, T, F, W >$$



☞ Ordinary net: weights equal to 1.



☞ A second perspective for net structures:

$$N = < P, T, Pre, Post >$$

Pre-incidence function, $Pre(p,t)$: $PxT \to \mathbb{N}^+$ ({0,1} for ordinary)

Post-incidence function, $Post(t,p)$: $PxT \to \mathbb{N}^+$ ({0,1} for ordinary)

☞ *Incidence matrices:*

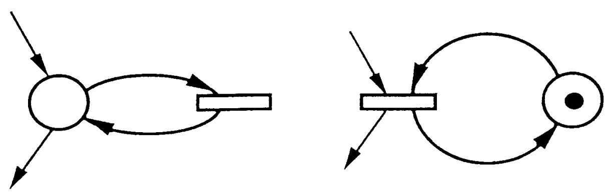
- Pre-incidence : C^-_{nxm} = Pre_{nxm}
- Post-incidence : C^+_{nxm} = Post_{nxm}
- Incidence : $C_{nxm} = C^+ - C^-$

where : $n = |P|$
$m = |T|$

$$C^- = \begin{array}{c|cccccc} & a & b & c & d & e & f \\ \hline p1 & 1 & 0 & 0 & 0 & 0 & 0 \\ p2 & 0 & 1 & 0 & 0 & 0 & 0 \\ p3 & 0 & 0 & 0 & 1 & 0 & 0 \\ p4 & 0 & 0 & 1 & 0 & 1 & 0 \\ p5 & 0 & 0 & 0 & 1 & 0 & 0 \\ p6 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

$$C^+ = \begin{array}{c|cccccc} & a & b & c & d & e & f \\ \hline p1 & 0 & 0 & 0 & 1 & 0 & 0 \\ p2 & 1 & 0 & 0 & 0 & 0 & 0 \\ p3 & 0 & 1 & 0 & 0 & 0 & 0 \\ p4 & 1 & 0 & 0 & 0 & 0 & 1 \\ p5 & 0 & 0 & 1 & 0 & 0 & 0 \\ p6 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$

☞ Incidence matrix "does not see" selfloops



☞ N is the *static* structure

M allows to represent the *dynamic* on N

☞ The *State Equation:*

- $M_{k-1} \; [t > M_k \Leftrightarrow M_k \; = M_{k-1} + C(t) =$

  $= M_{k-1} + C^+(t) - \bar{C}(t) \geq 0$

- Integrating along an execution (firing sequence):

$$M_0 \; [\; \sigma > M_k \Rightarrow M_k = M_0 + C \cdot \bar{\sigma}$$

where $\bar{\sigma}$ is the *firing count vector* of σ.

- Very important: Unfortunately

$$M_k \; = M_0 + C \cdot \bar{\sigma} \geq 0 \; , \quad \bar{\sigma} \geq 0 \quad \not\Rightarrow \quad M_0 \; [\; \sigma > M_k$$

- Therefore: convex geometry/linear algebra based analysis techniques cannot conclude always in general: *semidecision.*

2 Reduction of net models

2.1 Basic concept

☞ $< N^i, M_0^{\ i} > \ \rightarrow \ < N^{i+1}, M_0^{\ i+1} >$

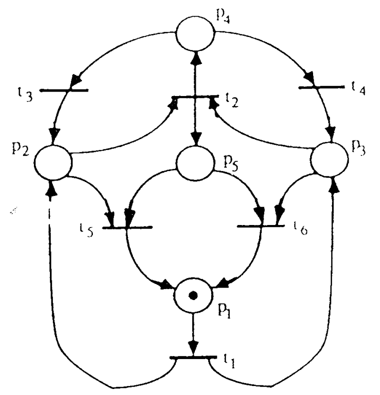
while:
- properties under study (e.g. boundedness, liveness,...) are *preserved*
- $< N^{i+1}, M_0^{\ i+1} >$ is *simpler* to analyse (e.g. smaller reachability graph).

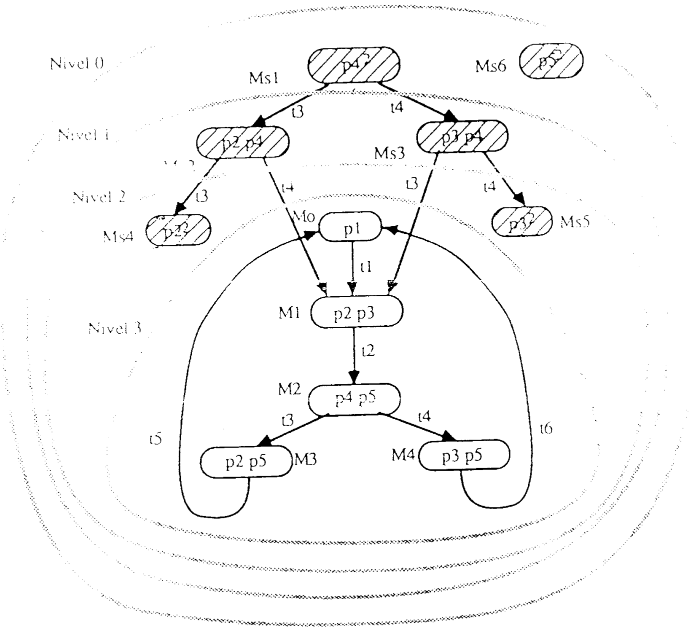
RULE:
- Structural precondition
- Marking precondition
- Structural change
- Marking change

APPLICATION: **if** preconditions are true (thus properties are preserved)

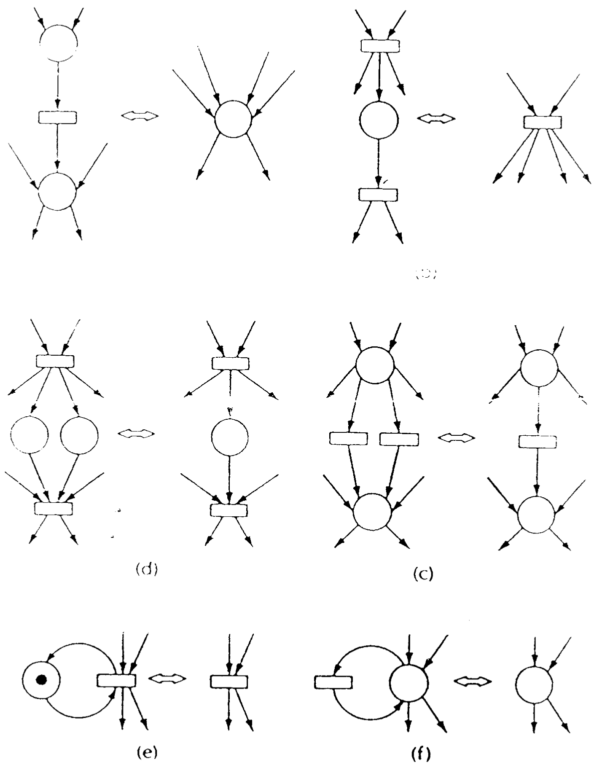
then make changes

PROBLEM:
- ∃ unreducible systems given a reduction kit.
- Tradeoff: kit reduction power *versus* kit complexity

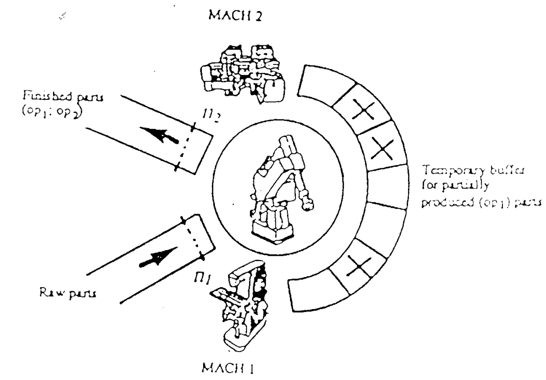


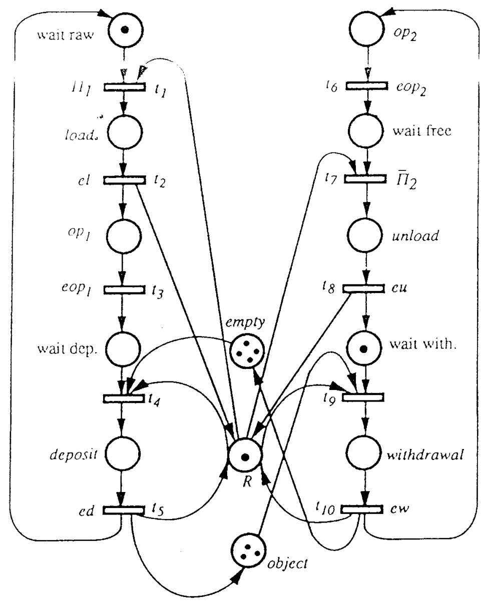


2.2 A very simple kit of reduction rules preserving liveness and boundedness



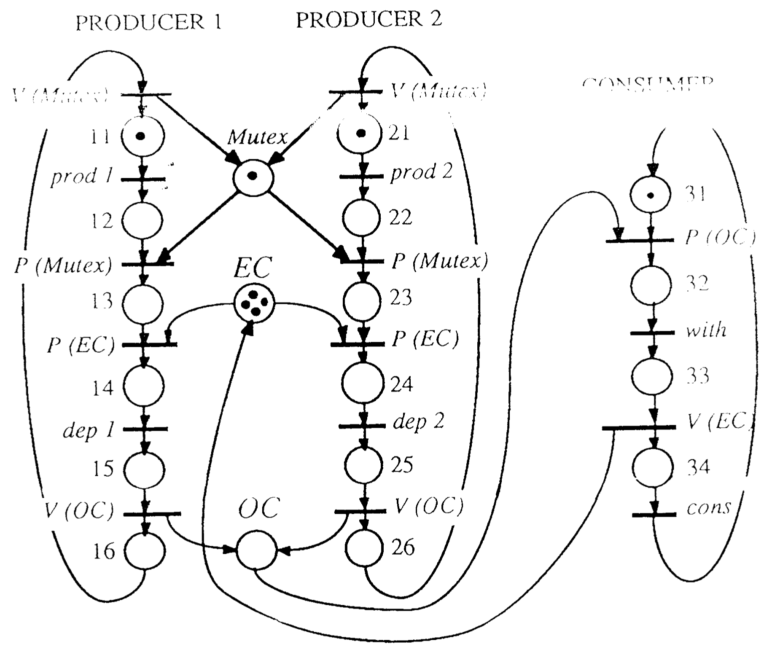
An illustrative example: A Producer-Consumer system with a Store operated under Mutual Exclusion

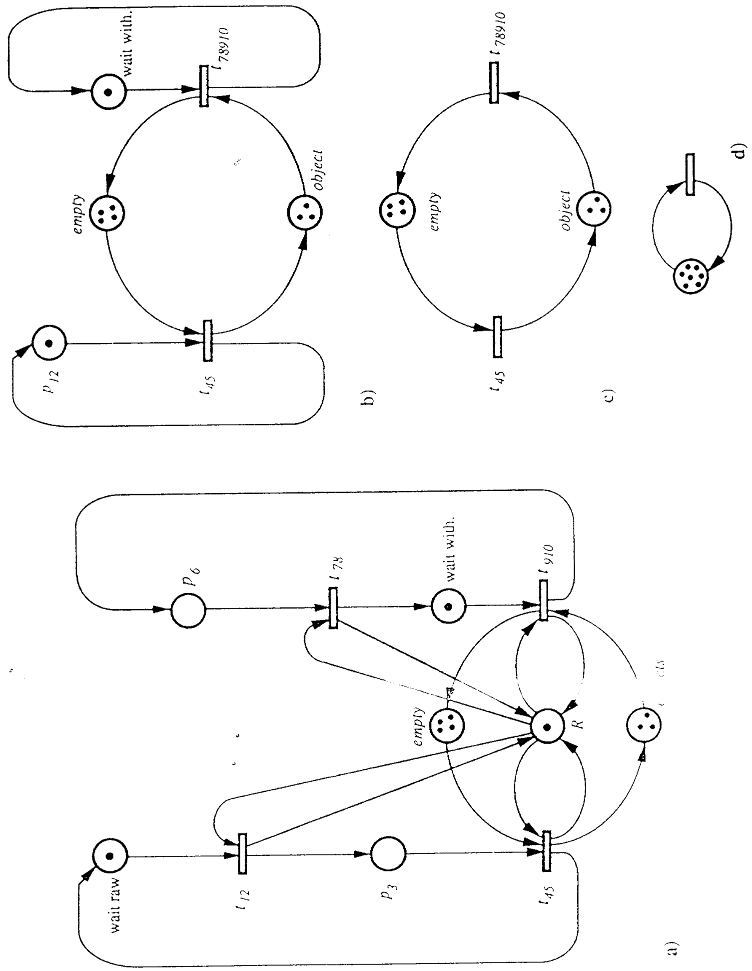




Two producers and the consumer with bounded buffer and mutual exclusion (*Note*: The initialization is not considered).

PRODUCER1	PRODUCER 2	CONSUMER
loop	*loop*	*loop*
produce 1	produce 2	P(OC)
P (Mutex)	P (Mutex)	P (OC)
P (EC)	P (EC)	V (EC)
deposit 1	deposit 2	withdraw
V (OC)	V (OC)	consume
V (Mutex)	V (Mutex)	*endloop*
endloop	*endloop*	





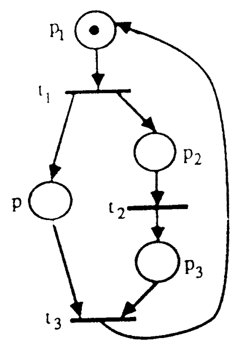
The reductio[illegible] process shows (see Fig. d) that the net system in Figure is live, 7-bounded and reversible

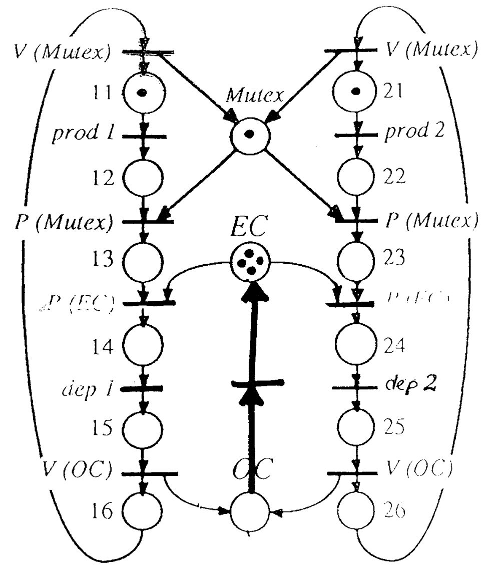
2.3 Implicit places: Basic concept and search technique

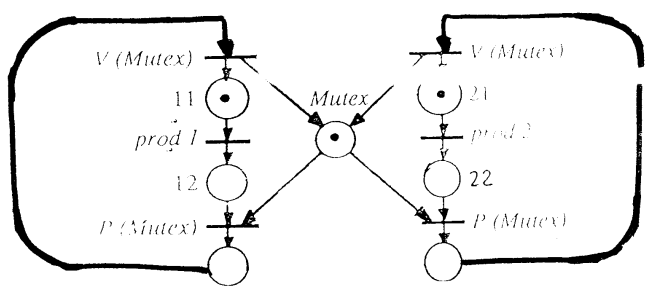
Definition: A place p is *implicit* in $<N, M_0>$ if never is the unique to constraint the firing of its output transitions.

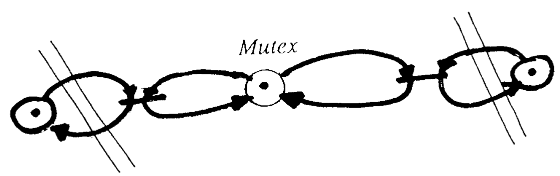
Therefore: Removing implicit places does not change the set of firable sequences.

thus: removing implicit places preserve liveness, synchronic properties (lead, distance, slack, fairness, ...)









Searching implicit places

---

(1) $v = \min\ Y^T . M_o$

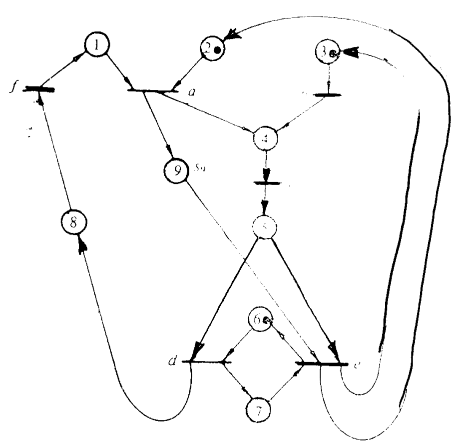
s. t. $C(p) = Y^T . C$

$Y \geq 0$

(2) **if** $M_o(p) \geq v$ **then** p *is implicit*

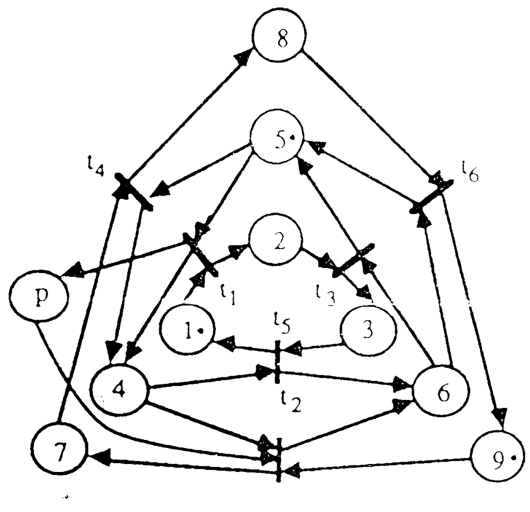
else ???

(a sufficient but not necessary condition)



Searching implicit places

Place p is not structurally implicit



Remark. Any structurally implicit place p (i. e. $C(p) = Y^T . C$, $Y \geq 0$) can be maded implicit just adding tokens (sooner or later v will be smaller or equal than $Y^T . Mo$)

2.4 Reduction on net subclasses can be complete: the free choice example

☞ Two reduction rules with orthogonal rôles:

R1) Reduce some *sequential* (SM) subnets into places

R2) Remove some places defining *concurrency*.

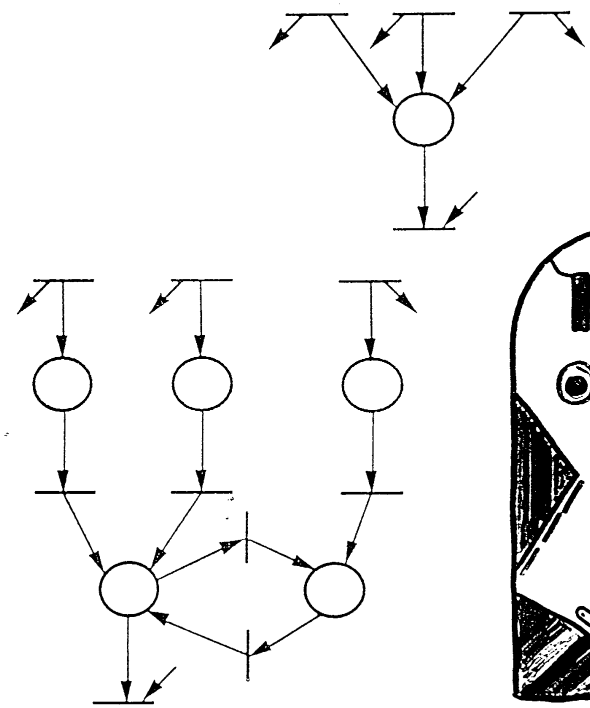
> R1) Is a particular case of the *macroplace* rule
>
> R2) Is a particular case of the *structurally implicit place* rule

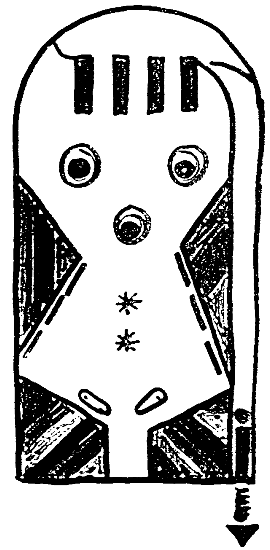
☞ R1 & R2 lead to:

- *SOUNDEDNESS*
- *COMPLETENESS*: All LBFC can be reduced to a marked *seed net* (single place-with a selfloop transition).

MACROPLACE

RULE 1: Places with $|p^{\bullet}|=1$ can be refined into FLIPPER-SUBNETS (i.e. sequential subnets with one single way-out).





☞
* Liveness &

* The bound of the net (⇒ boundedness)

are preserved.

MARKING IMPLICIT

p is a Marking Structurally Implicit place (MSIP) iff it is a positive linear combination of other places.

p is MSIP $\Leftrightarrow$ $\exists\, Y \gneq 0 \quad Y^T C = l_p$
where: * C is the incidence matrix
* l_p is the row representing p

RULE 2: MSIP with $|p^{\bullet}|=1$ (to preserve FC!) can be added if no new P-semiflow of the new net is unmarked.

- * Liveness &
- * Boundedness (but not the bound)

are preserved

Now: SL is an interesting property, but what about liveness for a given M_0 ?

A SL & SB (i.e. lively and boundely markable) FC net is

live for M_0 iff all p-invariants are marked by M_0

Equivalently:
non live $\Leftrightarrow \exists\, Y \geq 0\ ,\ Y^T{\cdot}C = 0\ \&\ Y^T{\cdot}M_0 = 0$

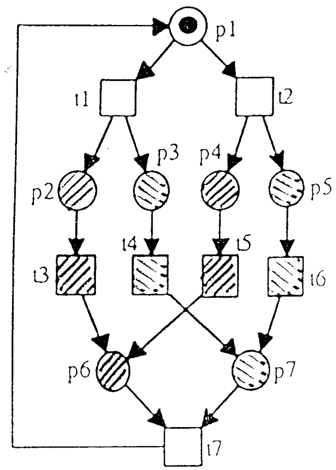
Therefore, for FC nets:

(1) SL & SB can be characterized in CG/LA terms.

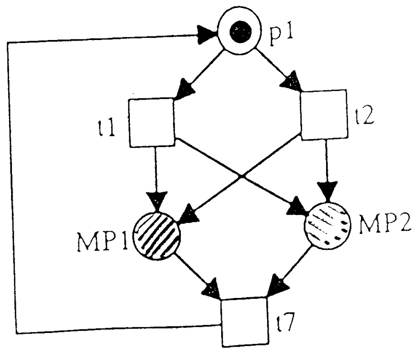
(2) Liveness in SL &SB FC nets can be also characterized in CG terms.

As $B \Leftrightarrow SB$ for live FC then:

L & B can be decided in polynomial time



N



3 Structure Theory: A Convex Geometry/ Linear Algebra perspective

☞ The key point: *The net state equation*

- $M_0 [\sigma > M \nRightarrow M = M_0 + C \cdot \bar{\sigma}$

  $M \in \mathbb{N}^n, \ \bar{\sigma} \in \mathbb{N}^m$

- Unfortunately the reverse is *not* true:

SPOURIOUS SOLUTIONS

☞ What does we undestand (today) about the behaviour of P/T systems using CG/LA?

- Some simple explanations & answers
- Some semidecision & decision techniques

☞ Structural:

- Boundedness: boundedness $\forall M_0$
- Liveness: $\exists M_0$ making live the system

3.1 Structural computation of the bound of a place. Structural boundedness characterization

$$SB(p) = \max\ M(p)$$
$$\text{subject to}\quad M = M0 + C \cdot \bar{\sigma}$$
$$M \geq 0,\quad \bar{\sigma} \geq 0$$

(Primal) LPP

☞ SB(p) can be computed in polynomial time

☞ The dual of the above problem: $M(p) = e_p \cdot M$

$$SB(p) = \min\ Y^T \cdot M_0$$
$$\text{subject to}\quad Y^T \cdot C \leq 0$$
$$Y \geq e_p$$

(Dual) LPP

☞ Duality in LP:

- The primal LPP has always a solution: $M = 0,\ \bar{\sigma} = 0$
- By *duality* the Dual LPP is:

  * *bounded* if the Primal LPP is bounded or
  * *non feasible* if the primal problem is unbounded.

p *is SB iff* $\exists Y \geq e_p,\quad Y^T \cdot C \leq 0\ (\Rightarrow Y^T \cdot M \leq Y^T \cdot M_0)$

N *is SB iff* $\exists Y \geq \mathbb{1},\quad Y^T \cdot C \leq 0$

3.2 Structural boundedness and structural liveness

3.2.1 Conservativeness and Consistency

☞ t is structurally repetitive iff $\exists M_0$ and $\bar{\sigma}$ such that

$$M_0 [\, \bar{\sigma} > M_0 \text{ and } \bar{\sigma} \geq e_t$$

☞ t is structurally repetitive iff the next LPP is unbounded

$$SR(t) = \max\ e_t \cdot \bar{\sigma}$$
$$\text{subject to } M = M_0 + C \cdot \bar{\sigma}$$
$$M \geq 0,\ \bar{\sigma} \geq 0$$

☞ Thus (through duality & boundednes in LP):

$$t \text{ is SR iff } \exists X \geq e_t \text{ such that } C \cdot X \geq 0$$

$$N \text{ is SR iff } \exists X \geq \mathbb{1} \text{ such that } C \cdot X \geq 0$$

☞ And it is possible to write

$$N \text{ is SB \& SL} \nRightarrow N \text{ is SB \& SR} \Leftrightarrow$$
$$\Leftrightarrow N \text{ is Cv \& Cv}$$

where: $N \text{ is Cv} \Leftrightarrow \exists Y \geq \mathbb{1},\ Y^T \cdot C = 0$

$N \text{ is Ct} \Leftrightarrow \exists X \geq \mathbb{1},\ C \cdot X = 0$

Summarizing

- $N \text{ is SB} \Leftrightarrow \exists Y \geq \mathbb{1},\quad Y^T \cdot C \leq 0$
- $N \text{ is SB \& SL} \nRightarrow \exists Y \geq \mathbb{1},\quad Y^T \cdot C = 0 \quad \{Cv\}$

  $\exists X \geq \mathbb{1},\quad C \cdot X = 0 \quad \{Ct\}$

3.2.2 *The Rank Property*

☞ It is po...ible to improve the knowledge about SL in SB nets?

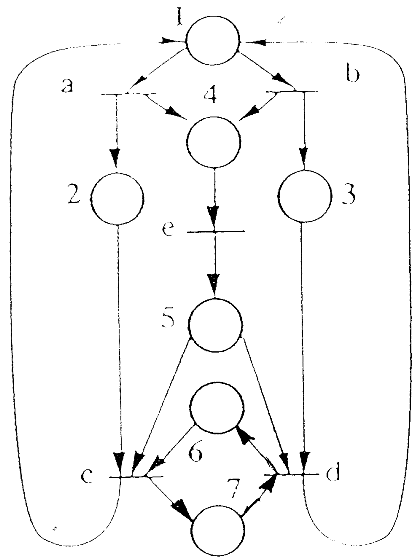
Definition: t_a and t_b are in *equality conflict relation* (ECR) iff $Pre(t_a) = Pre(t_b)$

☞ ECR is an *Equivalence* Relation:

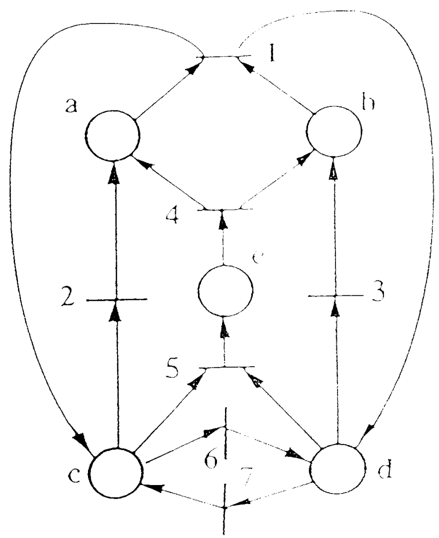
Let:
- D_i be an equivalence class
- $\delta_i = |D_i| - 1$
- $\delta = \Sigma_i \delta_i$

Property: If N is SB & SL then:

- N is Cv & Ct
- $rank(C) \leq |T| - 1 - \delta$



a)



b)

$rank(C) = 4$
$|T|-1-\delta = 5 - 1 - 1 = 3$
4
SL? N is not struct-Live

$rank(C) = 4$
$|T| - 1 - \delta = 7 - 1 - 2 =$

SL? No answer

3.3 Boundedness and liveness for free choice systems. Some consequences.

Property: FC Nets: $L \,\&\, B \not\Rightarrow SL \,\&\, SB$

Remark: For general live P/T nets B does not imply SB.

☞ $SL \,\&\, SB$ is characterizable for FC nets

The FC net N is SL & SB iff • N *is* $C_t \,\&\, C_v$ • $rank\,(C) = m - 1 - \delta = m - 1 - (a - n)$ *where* $a = \#$ *arcs in Pre*

Some consequences for FC nets:

(1) SL & SB can be computed in *polynomial time*

(2) Hack's *Duality Theorem*:

Let N be a FC net and N_{rd} its reverse-dual

N *is* $SL \,\&\, SB \Leftrightarrow N_{rd}$ *is* $SL \,\&\, SB$

(3) Soundedness of the primal and dual complete reduction kits for LBFC

Concluding Remarks

☞ A number of analysis techniques for P/T nets have been presented:

- property preserving *reduction* of nets
- *convex geometry / linear algebra*
- *graph* theoretical arguments

☞ For distinguished *net subclasses* efficient analysis algorithms are available.

☞ In general: The analysis problem should be considered using different analysis techniques in a *COOPERATIVE* way.

☞ Interleaving of *FUNCTIONAL* and *PERFORMANCE* (structural) analysis.

AN INTRODUCTION TO GENERALIZED STOCHASTIC PETRI NETS

Gianfranco Balbo

Dipartimento di Informatica
Università di Torino

Outline

- **TIMED PETRI NETS**
  - Timed places, tokens, arcs, transitions
  - Race and preselection
  - Memory
  - Single and multiple server semantics
- **STOCHASTIC PETRI NETS**
  - The exponential distribution
  - Markov chains
  - Isomorphism between SPNs and MCs
  - Example
- **GENERALIZED SPNs**
  - Immediate transitions and priority
  - GSPN definition
  - Extended conflict sets
  - Isomorphism between GSPNs and MCs
  - Performance indices
  - Example
- **AN APPLICATION OF GSPNS**
- **CONCLUSIONS**

Prerequisites

- **The basic definitions of Petri net theory**
  - places, transitions, arcs, tokens
  - marking
  - enabling, firing, reachability
  - (enabling degree)
  - conflict, confusion
  - (invariants)
- **Some elementary notions of probability theory**
  - random variable
  - stochastic process
  - pdf, PDF
  - state space
  - averages
  - sojourn times
  - (ergodicity)
  - (Little's formula)

Items in parentheses are optional.

TIMED PETRI NETS

Timing specifications

Time is introduced in Petri nets to model the interaction among several activities considering their starting and completion times

The introduction of time specifications corresponds to an interpretation of the model by means of

- observation of the autonomous (untimed) model
- definition of a non-autonomous model

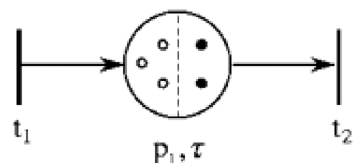
Time specifications should provide

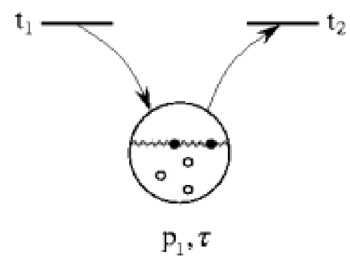
- consistency among autonomous and non-autonomous models
- non-determinism reduction on the basis of time considerations
- support for the computation of performance indices

Timed places

Several approaches are possible for the introduction of temporal specifications in PN models:

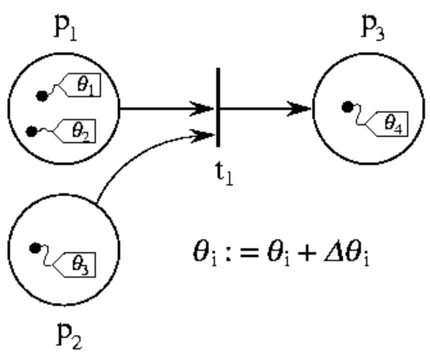
- time may be associated with places (TPPN):
  - tokens generated in an output place become available to fire a transition only after a delay has elapsed; the delay is an attribute of the place





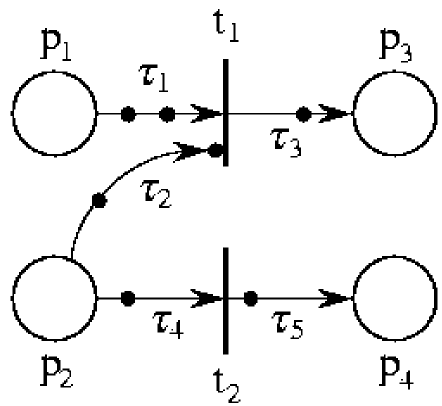
Timed tokens

- time may be associated with tokens:
  - tokens carry a time stamp that indicates when they are available to fire a transition; this time stamp can be incremented at each transition firing.



Timed arcs

- time may be associated with arcs:
  - a travelling delay is associated with each arc; tokens are available for firing only when they reach a transition



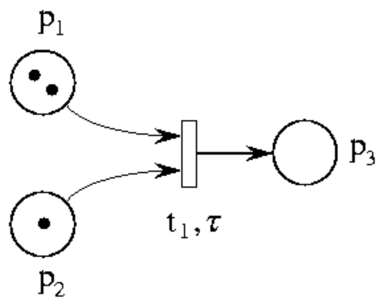
Timed transitions

- time may be associated with transitions (TTPN); transitions represent activities
  - activity start corresponds to transition enabling,
  - activity end corresponds to transition firing

Different firing policies may be assumed:

- three-phase firing
  1. tokens are consumed from input places when the transition is enabled
  2. the delay elapses
  3. tokens are generated in output places

- atomic firing
  tokens remain in input places for the transition delay; they are consumed from input places and generated in output places when the transition fires



Atomic firing

We shall consider TTPN with atomic firing.

TTPN with atomic firing can preserve the basic behaviour of the underlying untimed model.

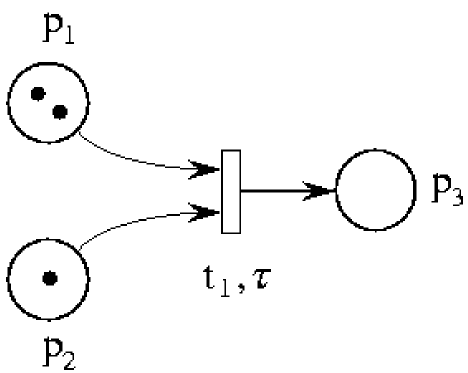
It is thus possible to qualitatively study TTPN with atomic firing exploiting the theory developed for untimed (autonomous) PN (reachability set, invariants, etc.).

Timing specifications may affect the qualitative behaviour of the PN when they describe *constant* and *interval* firing delays.

Internal timer

We can explain the behaviour of one timed transition with atomic firing by assuming that it incorporates a timer.

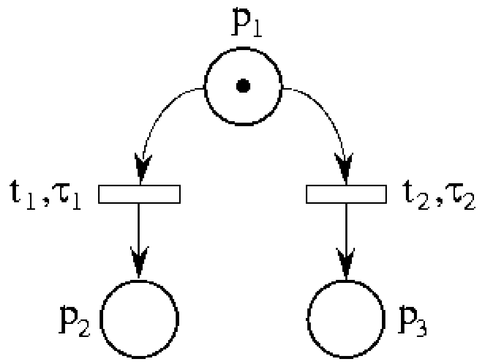
- When the transition is enabled, its timer is set to the current delay value
- Then, the timer is decremented at constant speed, until it reaches the value zero
- At this point the transition fires



Conflicts

When more than one timed transition with atomic firing is enabled, the behaviour is similar, but a problem arises:

Which one of the enabled transitions is going to fire?

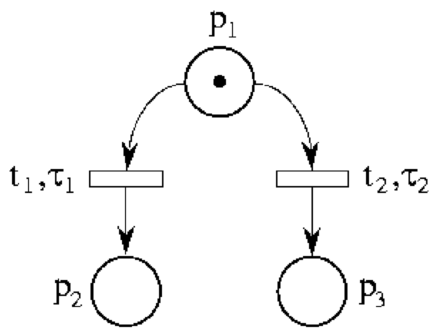


Selection rules

Two alternative selection rules:

- **preselection:**
  the enabled transition that will fire is chosen when the marking is entered, according to some metric (priority, probability, ...)

- **race:**
  the enabled transition that will fire is the one whose firing delay is minimum

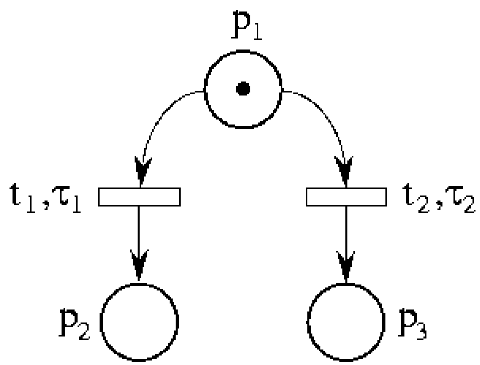


Memory policies

When a timed transition is disabled by a conflicting transition, a problem arises:

How is the transition timer set when the transition will again become enabled?

How does the transition keep **memory** ***of its past enabling time?***



Basic mechanisms

Two basic mechanisms can be defined:

- Continue:
  the timer associated with the transition holds the present value and will *continue* later on the count-down

- Restart:
  the timer associated with the transition is *restarted*, i.e., its present value is discarded and a new value will be generated when needed

Transition memory policies

From the two basic mechanisms it is possible to construct several transition memory policies; the usual ones are:

- Resampling:
  - At each and every transition firing, the timers of all timed transitions in the timed PN system are discarded (restart mechanism).
  - No memory of the past is recorded.
  - After discarding all timers, new values of the timers are set for the transitions that are enabled in the new marking.

- **Enabling memory:**
  - **At each transition firing, the timers of all timed transitions that become disabled are restarted, whereas the timers of all timed transitions that remain enabled hold their present value (continue mechanism).**
  - **The memory of the past is recorded with an *enabling memory variable* associated with each transition.**
  - **The enabling memory variable accounts for the work performed by the activity associated with the transition since the last instant of time when its timer was set.**

    The enabling memory variable measures the enabling time of the transition since the last instant of time it became enabled.

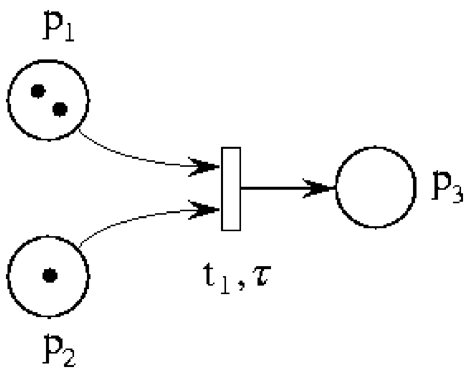
- **Age memory:**
  - **At each transition firing, the timers of all timed transitions hold their present values (continue mechanism).**
  - **The memory of the past is recorded with an *age memory variable* associated with each timed transition.**

    The age memory variable accounts for the work performed by the activity associated with the transition since the time of its last firing.
  - **The age memory variable measures the *cumulative* enabling time of the transition since the last instant of time when it fired.**

Transition enabling

The *enabling degree* of a transition is the number of times the transition could fire in the given marking before becoming disabled.

When the enabling degree of a transition is > 1, attention must be paid to the timing semantics.



Server semantics

Three cases are common:

- Single-server semantics
- Infinite-server semantics
- Multiple-server semantics

Single-server semantics:

A firing delay is set when the transition is first enabled, and new delays are generated upon transition firing if the transition is still enabled in the new marking.

Enabling sets of tokens are processed *serially* and the temporal specification associated with the transition is independent of the enabling degree;

Infinite-server semantics:

Every enabling set of tokens is processed as soon as it forms in the input places of the timed transition.

Its corresponding firing delay is generated at this time, and the timers associated with all these enabling sets run down to zero concurrently.

Multiple enabling sets of tokens are thus processed *in parallel*.

The overall temporal specifications of transitions with this semantics depend directly on their enabling degrees

Multiple-server semantics:

Enabling sets of tokens are processed as soon as they form in the input places of the transition up to a maximum degree of parallelism (say K).

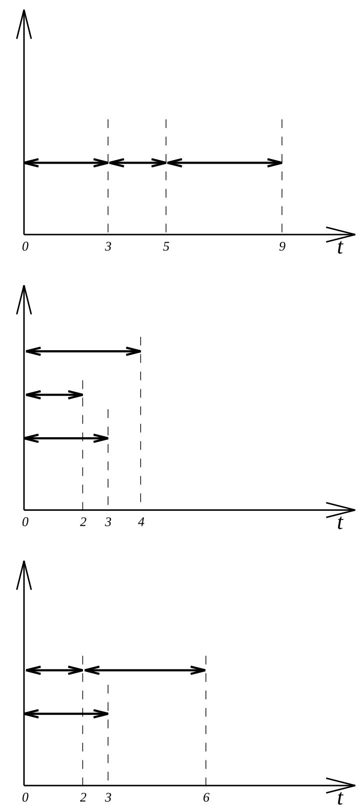
For larger values of the enabling degree, the timers associated with new enabling sets of tokens are set only when the number of concurrently running timers decreases below the value K.

The overall temporal specifications of transitions with this semantics depend directly on their enabling degrees up to a threshold value K

Example of server semantics

Consider a timed transition with enabling degree equal to 3.

The three enablings are associated with firing delays equal to 3, 2, and 4 time units.



Queueing policies

Upon firing of a transition, input tokens are removed at random

Specific queueing policies must be explicitly represented at model level

Firing and selection rules

We consider TTPN with atomic firing and race selection rule.

Transitions within one TTPN can use

- Resampling
- Enabling memory
- Age memory

and

- Single-server semantics
- Infinite-server semantics
- Multiple-server semantics

in any combination

STOCHASTIC PETRI NETS

Probabilistic interpretation

Timed Transition PN with atomic firing in which all transition delays are *random variables* with negative exponential distributions are called *Stochastic PN (SPN)*.

The dynamic behaviour of a SPN is described through a *stochastic process*.

Definitions

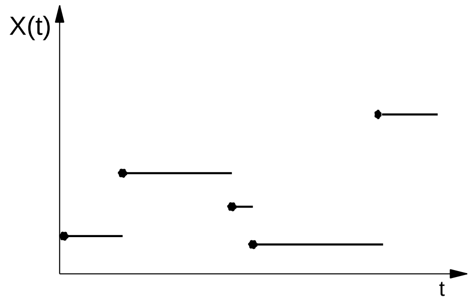
A ***random variable*** is a real function defined over a probability space.

Stochastic processes are mathematical models useful for the description of phenomena of a probabilistic nature as a function of a parameter that usually has the meaning of time.

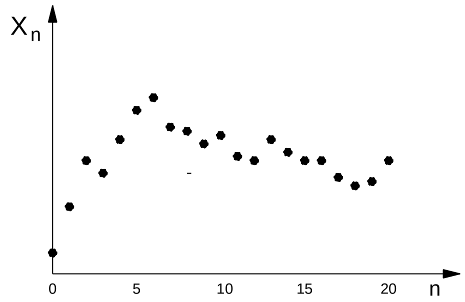
A stochastic process $\{X(t), t \in T\}$ is a family of random variables defined over the same probability space, indexed by the parameter t and taking values in the state space S.

Stochastic processes

A *sample path* (or realization) of a stochastic process is a function of time.



Continuous time



Discrete time

The probabilistic description of a random variable X is given by its *probability density function* **(pdf)**

$$f_X(x) = \frac{d}{dx} P\{X \leq x\} \qquad -\infty < x < \infty$$

The probabilistic description of a random process is given by the joint pdf of any set of random variables extracted from the process.

$$P\{X(t_1) \leq x_1, X(t_2) \leq x_2, ..., X(t_n) \leq x_n\}$$

In the general case the complete probabilistic description of a random process is not feasible.

MARKOVIAN processes **are one special class of stochastic processes for which the probabilistic description is simpler and of particular relevance.**

A process that satisfies the Markov property:

$$P\{X(t) \leq x | X(t_n) = x_n, X(t_{n-1}) = x_{n-1}, \cdots$$
$$X(t_0) = x_0\} = P\{X(t) \leq x | X(t_n) = x_n\}$$

with $t > t_n > t_{n-1} > \cdots > t_0$ is called a *Markovian process*.

If the state space is denumerable, the process is a *Markov chain*.

If the parameter t is continuous, the process is a *continuous-time* Markov chain (CTMC).

A continuous-time Markov chain (CTMC) is a stochastic process where

- sojourn times in states are exponentially distributed random variables

- the future evolution depends only on the present state, not on the past history

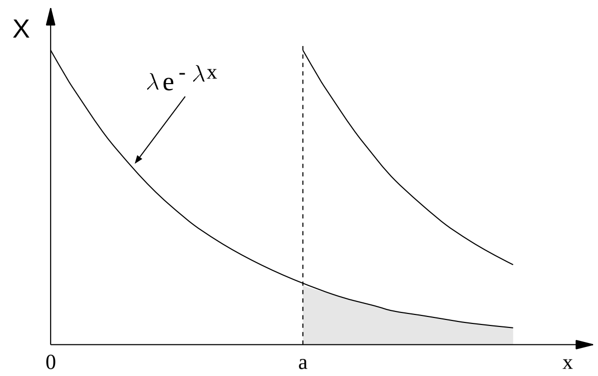
Exponential distributions

The exponential pdf

$$f_X(x) = \lambda e^{-\lambda x} \qquad (x \geq 0)$$

is the only continuous pdf for which the memoryless property holds:

$$P\{X > x + \alpha | X > \alpha\} = P\{X > x\}$$



normalization

The exponential pdf

$$f_X(x) = \lambda e^{-\lambda x} \qquad (x \geq 0)$$

is defined only by its *rate* λ, which is the inverse of its average value:

$$E[X] = \frac{1}{\lambda}$$

Given two random variables X and Y with exponential pdf

$$f_X(x) = \lambda e^{-\lambda x} \qquad (x \geq 0)$$

$$f_Y(y) = \mu e^{-\mu y} \qquad (y \geq 0)$$

the new random variable $Z = \min(X, Y)$ also has an exponential pdf

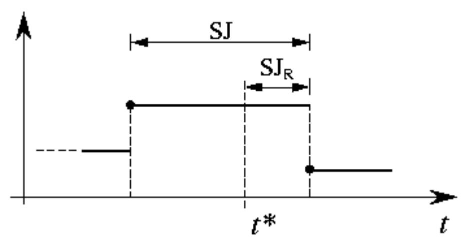
$$f_Z(z) = (\lambda + \mu) e^{-(\lambda+\mu)z} \qquad (z \geq 0)$$

In fact,

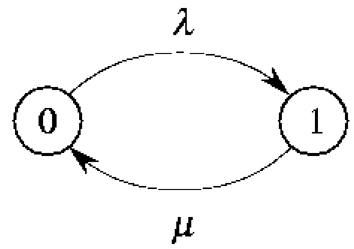
$$\begin{aligned} F_Z(z) &= 1 - Pr\{Z > z\} \\ &= 1 - Pr\{X > z, Y > z\} \\ &= 1 - e^{-\lambda z} e^{-\mu z} = 1 - e^{-(\lambda+\mu)z} \qquad (z \geq 0) \end{aligned}$$

Markov chains

The residual sojourn time in a state of a Markov chain is a random variable with the same distribution as the whole sojourn time.



A CTMC can be described through a *state transition rate diagram*, or equivalently with a *state transition rate matrix*, also called *infinitesimal generator*, denoted by $\boldsymbol{Q}$.



$$\boldsymbol{Q} = \begin{bmatrix} -\lambda & \lambda \\ \mu & -\mu \end{bmatrix}$$

The solution of a CTMC at time t is the probability distribution over the set of states:

$$\boldsymbol{\pi}(t) = (\pi_1(t), \pi_2(t), \pi_3(t), \cdots)$$

with

$$\pi_i(t) = P\{X(t) = i\}$$

It can be proven that

$$\frac{d\boldsymbol{\pi}(\tau)}{d\tau} = \boldsymbol{\pi}(\tau)\boldsymbol{Q}$$

whose solution can be formally written as

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(0)\boldsymbol{H}(t)$$

with

$$\boldsymbol{H}(t) = e^{\boldsymbol{Q}t}$$

This is a very elegant solution that is however usually very expensive to compute since the matrix exponentiation is defined by the following infinite sum

$$e^{\boldsymbol{Q}\tau} = \sum_{k=0}^{\infty} \frac{(\boldsymbol{Q}\tau)^k}{k!}$$

The solution of a CTMC at steady-state is the probability distribution over the set of states.

The steady-state distribution exists for *ergodic* CTMCs.

The steady-state distribution

$$\boldsymbol{\pi} = (\pi_1, \pi_2, \pi_3, \cdots)$$

with

$$\pi_i = \lim_{t \to \infty} P\{X(t) = i\}$$

is computed as the solution of the linear system of equations

$$\boldsymbol{\pi Q} = \mathbf{0}$$

with the normalizing condition

$$\sum_i \pi_i = 1$$

Definition of stochastic Petri nets

Formally, an SPN is defined through an 8-tuple:

$$\mathrm{SPN} = (P, T, I(.), O(.), H(.), W(.), M_0)$$

where

- $\mathrm{PN} = (P, T, I(.), O(.), H(.), M_0)$ **is the marked PN underlying the SPN**
- $W(.)$ **is a function defined on the set of transitions that associates a rate with each transition. This rate is the inverse of the average firing time of the transition**

SPNs can be proved to be isomorphic to CTMCs: the reachability graph of the SPN corresponds to the state transition rate diagram of the MC.

This can be easily seen in the case of simple subclasses of Petri nets such as: ***Finite State Machines*** **and** ***Marked Graphs***

SPNs without choices and synchronizations

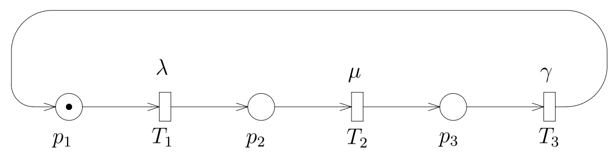
- The net has the structure of both a Finite State Machine (no transition has more than one input and one output place) and of a Marked Graph (no place has more than one input and one output transition);
- the initial marking contains only one token

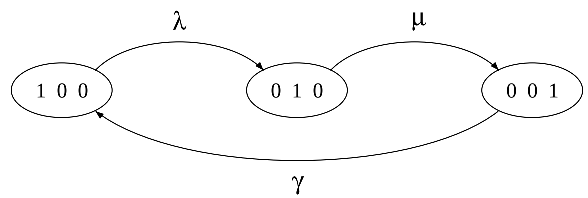
Each place of the net univocally identifies a state of the net.

Each place of the net maps into a state of the corresponding probabilistic model.

The time spent by the token in each of its places is completely determined by the characteristics of the only transition that can withdraw it from that place.

The probabilistic model that represents the behaviour of the net (***Marking Process***) is a CTMC





SPNs with choices

- The net has the structure of a Finite State Machine (no transition has more than one input and one output place);

- the initial marking contains only one token

Conflicts arise when several transitions share a common input place

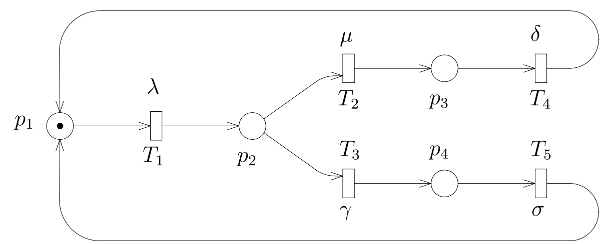
A race starts among the simultaneously enabled transitions.

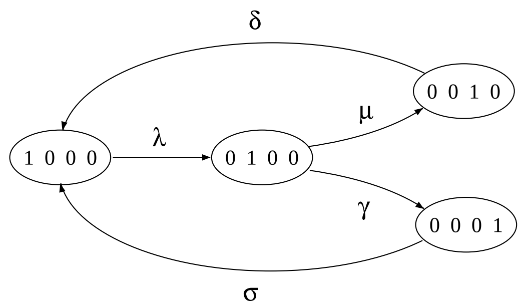
The race is won by one of the transitions and the way of dealing with the partially completed activities of the transitions that were interrupted becomes an issue (in general).

When the firing times of the transitions of the net have negative exponential distributions, their memoryless property makes the distinction among

- resampling
- enabling memory
- age memory

irrelevant and the CTMC corresponding to the SPN is obtained from the net in a straightforward manner.

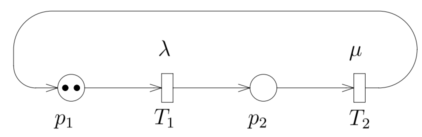




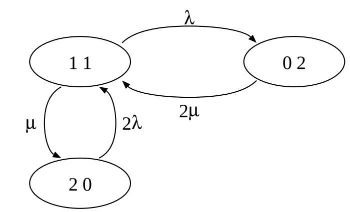
More complex situations arise already when several tokens are allowed in the initial markings of these simple models.

- Service semantics adopted when the input place of a transition contains several tokens;
- Queueing policy assumed with respect to the tokens residing in the input place of a transition.

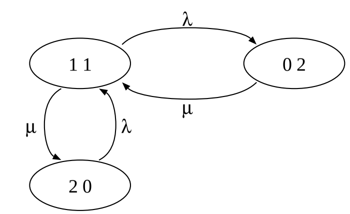
A simple example



IS (on both transitions)



SS (on both transitions)



Queueing policy

It is possible to show that when the firing times are exponentially distributed and the performance figures of interest are only related to the moments of the number of tokens in the input place of a transition many queueing policies yield the same results and thus the *random order* (that is the most natural in the Petri net context) can be assumed.

In general, the CTMC associated with a given SPN system is obtained by applying the following simple rules:

1. The CTMC state space $S = \{s_i\}$ corresponds to the reachability set $RS(\boldsymbol{m}_0)$ of the PN associated with the SPN $(\boldsymbol{m}_i \leftrightarrow s_i)$.
2. The transition rate from state s_i (corresponding to marking $\boldsymbol{m}_i$) to state s_j $(\boldsymbol{m}_j)$ is obtained as the sum of the firing rates of the transitions that are enabled in $\boldsymbol{m}_i$ and whose firings generate marking $\boldsymbol{m}_j$.

Assuming that all the transitions of the net operate with a single-server semantics and marking-independent speeds, and denoting with

- Q the *infinitesimal generator*,
- w_k the firing rate of T_k,
- $\boldsymbol{e_j}(\boldsymbol{m}_i) = \{h : T_h \in \boldsymbol{e}(\boldsymbol{m}_i) \wedge \boldsymbol{m}_i[T_h\rangle \boldsymbol{m}_j\}$ the set of transitions that bring the net from $\boldsymbol{m}_i$ to $\boldsymbol{m}_j$,

the components of Q are:

$$q_{ij} = \begin{cases} \sum_{T_k \in \boldsymbol{e_j}(\boldsymbol{m}_i)} w_k & i \neq j \\ -q_i & i = j \end{cases}$$

where

$$q_i = \sum_{T_k \in \boldsymbol{e}(\boldsymbol{m}_i)} w_k$$

Performance indices

The steady-state distribution $\boldsymbol{\pi}$ is the basis for a quantitative evaluation of the behaviour of the SPN expressed in terms of performance indices.

These results can be computed using a unifying approach in which proper index functions (also called ***reward functions***) are defined over the markings of the SPN and an average reward is derived using the steady-state probability distribution of the SPN.

Assuming that $r(\boldsymbol{m})$ represents one of such reward functions, the average reward can be computed using the following weighted sum:

$$R = \sum_{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0)} r(\boldsymbol{m}_i)\ \pi_i$$

Probability of a particular condition $\Upsilon(\boldsymbol{m})$ ***of the SPN.***

Define the following reward function:

$$r(\boldsymbol{m}) = \begin{cases} 1 & \Upsilon(\boldsymbol{m}) = true \\ 0 & otherwise \end{cases}$$

The desired probability is computed using the following expression:

$$P\{\Upsilon\} = \sum_{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0)} r(\boldsymbol{m}_i)\ \pi_i = \sum_{\boldsymbol{m}_i \in A} \pi_i$$

where $A = \{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0) : \Upsilon(\boldsymbol{m}_i) = true\}$.

Expected value of the number of tokens in a given place.

In this case the reward function is:

$$r(\boldsymbol{m}) \ = \ n \ \ \textbf{iff} \ \ m(p_j) \ = \ n$$

The expected value of the number of tokens in p_j is given by:

$$\boldsymbol{e}[m(p_j)] \ = \sum_{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0)} r(\boldsymbol{m}_i) \ \pi_i \ = = \ \sum_{n>0} \ [n \ P\{A(j,n)\}]$$

where $A(j,n) \ = \ \{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0) : \boldsymbol{m}_i(p_j) \ = \ n\}$ and the sum is obviously limited to values of $n \leq k$;
if the place is k-bounded.

Mean number of firings per unit of time of a given transition.

A transition may fire only when it is enabled, thus the reward function assumes the following form:

$$r(\boldsymbol{m}) \ = \begin{cases} w_j & T_j \in \boldsymbol{e}(\boldsymbol{m}) \\ 0 & otherwise \end{cases}$$

The mean number of firings of T_j per unit of time is then given by:

$$f_j \ = \sum_{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0)} r(\boldsymbol{m}_i) \ \pi_i \ = = \sum_{M_i \in A_j} w_j \ \pi_i$$

where $A_j \ = \ \{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0) : T_j \in \boldsymbol{e}(\boldsymbol{m}_i)\}$.

The average steady-state delay spent in traversing a sub-network can be computed from Little's formla

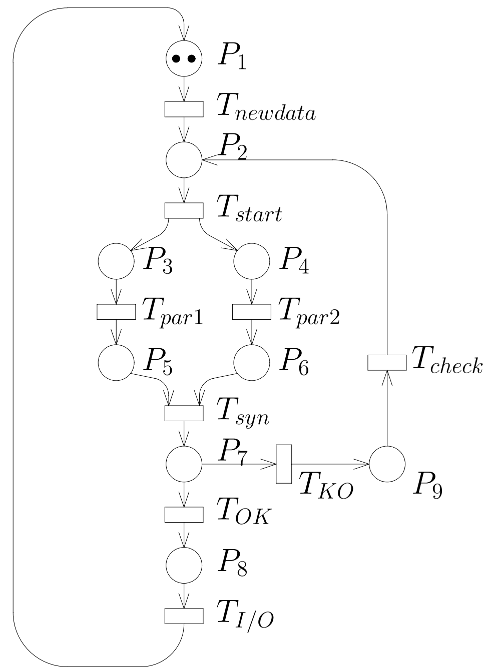
$$E[T] = \frac{E[N]}{E[S]}$$

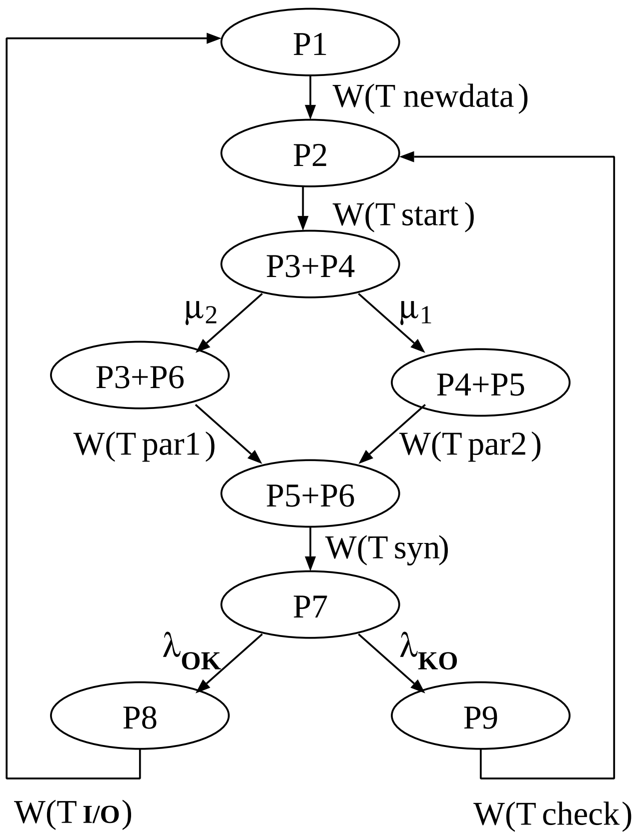
where $E[N]$ is the average number of (equivalent) tokens in the subnet, and $E[S]$ is the average input rate into the subnet.

Delay distributions are in general difficult to compute.

Example

The SPN description of a simple parallel system





State space for M(P1) = 1

Computation of μ_1**,** μ_2**:**

- **Total rate out of P3 + P4 is:**

$$W(T_{par1}) + W(T_{par2})$$

- **With what probability** T_{par1} **is the first to fire?**

$$\frac{W(T_{par1})}{W(T_{par1}) + W(T_{par2})}$$

- **Therefore:**

$$\begin{aligned} \mu_1 &= (W(T_{par1}) + W(T_{par2}))\frac{W(T_{par1})}{W(T_{par1}) + W(T_{par2})} \\ &= W(T_{par1}) \end{aligned}$$

Computation of λ_{OK}, λ_{KO}:

same computation as before:

$$\lambda_{OK} = W(T_{OK})$$

$$\lambda_{KO} = W(T_{KO})$$

but . . .

what is the meaning of $W(T_{OK})$ and $W(T_{KO})$?

→ **check activity: 0.0001**

- → **rate of** $10,000$
- → **probability of** OK/KO **is** 99% **vs.** 1%
- → $W(T_{OK}) = 9,900$ **and** $W(T_{KO}) = 100$

Parameter specifications:

transition	rate	value	semantics
$T_{newdata}$	λ	**1**	**infinite-server**
T_{start}	τ	**1000**	**single-server**
T_{par1}	μ_1	**10**	**single-server**
T_{par2}	μ_2	**5**	**single-server**
T_{syn}	σ	**2500**	**single-server**
T_{OK}	α	**9900**	**single-server**
T_{KO}	β	**100**	**single-server**
$T_{I/O}$	ν	**25**	**single-server**
T_{check}	θ	**0.5**	**single-server**

The consistency check operation has an average duration 0.0001 time units, and results in a success 99% of the times, and in a failure 1% of the times.

Performance indices:

- **Throughput of transition $T_{I/O}$:**
  - **1.504 success/time_units**
- **Average number of items under test:**
  - **0.031**
- **Average production time:**
  - **0.33 time_units**

GENERALIZED STOCHASTIC PETRI NETS

Two classes of transitions exist in GSPNs:

- *timed* transitions, whose delays are exponentially distributed random variables (like in SPNs)
- *immediate* transitions, whose delays are deterministically zero

Immediate transitions have been introduced in the model

- to account for instantaneous actions (e.g. choice among classes of clients);
- to implement specific modelling features (e.g. to empty a place);
- to account for time scale differences (e.g. bus arbitration and I/O accesses).

Immediate transitions have priority over timed transitions.

Several priority levels for immediate transitions can be defined. Immediate transitions at priority level n are called n-immediate.

The autonomous model associated with a GSPN is a ***Petri net with priorities***.

A transition t is said to have ***concession*** in marking M iff $M \geq I(t) \wedge M < H(t)$.
A transition t_j is defined to be enabled in markijg M iff it has concession in M and $\forall t_k \in T$ that have concession in M, $\pi_j \geq \pi_k$.

Effects induced by the presence of priorities

$$\Sigma \quad \Longleftrightarrow \quad \Sigma_\pi$$

- Properties

  → safety (invariant): must hold in all states

  → eventuality (progress): must hold in some state

- $RS(\Sigma) \supseteq RS(\Sigma_\pi)$

  → safety properties are maintained (absence of deadlocks, boundedness, mutual exclusion, ...)

  → eventuality properties are not necessarily maintained (reachability, liveness, ...)

Effects induced by the presence of priorities

$$\Sigma \quad \Longleftrightarrow \quad \Sigma_\pi$$

- Reachability

  → $M \in RS(\Sigma) \not\Rightarrow M \in RS(\Sigma_\pi)$

  but

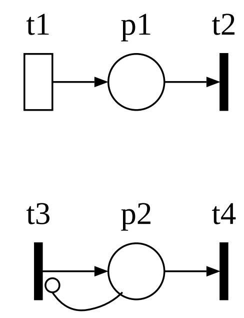
  → $M \in RS(\Sigma_\pi) \Rightarrow M \in RS(\Sigma)$

- Boundedness

  → Σ bounded $\Rightarrow \Sigma_\pi$ bounded

  but

  → Σ not bounded $\not\Rightarrow \Sigma_\pi$ not bounded

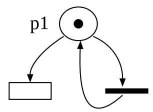


- Liveness - home states

  → priority can introduce or remove liveness

Some fine points:

1. **Need for priority**

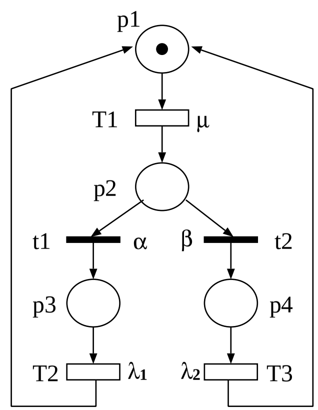


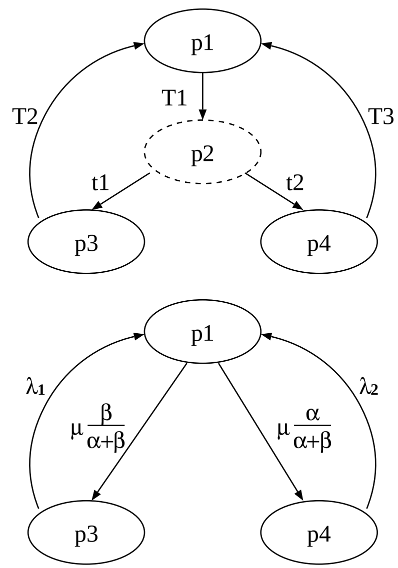
2. **Irrelevance of distinction between resampling - enabling - age due to the memoryless property of exponential distributions**

3. **Impossibility of two timers to expire at the same time**

   probability of extracting a specific sample x is equal to zero

Markings that enable timed transitions only are said to be *tangible*, whereas markings that enable n-immediate transitions are said to be *vanishing*.





Definition of generalized stochastic Petri nets

Formally, a GSPN is an 8-tuple:

$$\mathrm{GSPN} = (P, T, \Pi(.), I(.), O(.), H(.), W(.), M_0)$$

where

- $\mathrm{PN}_\pi = (P, T, \Pi(.),\ I(.), O(.),\ H(.),\ M_0)$ **is the marked PN with priority underlying the GSPN**
- $W(.)$ **is a function defined on the set of transitions**

The subnets formed by n-immediate transitions must be confusion-free.

The function $W(.)$ allows the definition of the stochastic component of a GSPN model. In particular, it maps transitions into real positive numbers.

The quantity $W(t_k) = w_k$ is called

- **the "rate" of transition t_k if t_k is timed**
- **the "weight" of transition t_k if t_k is n-immediate**

Rates are used like in SPNs.

Weights are used for the probabilistic resolution of conflicts of immediate transitions.

When a tangible marking is entered, the timed transitions that become enabled for the first time since their last firing, sample a firing delay instance and set their timer to the sampled value.

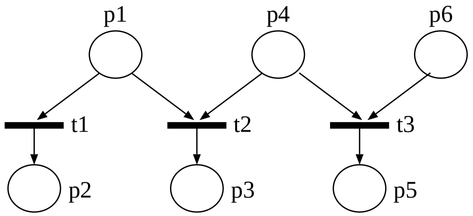
Then, all timers of the enabled (timed) transitions are decremented at equal speed, until one of them reaches the value zero.

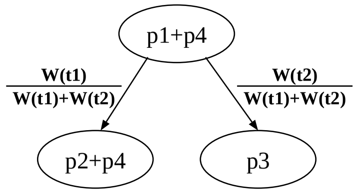
At this point the transition whose timer reached zero fires.

All the transitions that did not fire keep their timer readings, and their timers will be again decremented in the next marking in which the transition is enabled.

(Enabling memory was used in the description, but is irrelevant)

When a vanishing marking is entered, the weights of the enabled n-immediate transitions are used to probabilistically select the (n-immediate) transition(s) to fire. The time spent in any vanishing marking is deterministically equal to zero.





$W(t_1) = 10$ $W(t_2) = 20$ $W(t_3) = 44$

***note*:**

- **The weight of t_2 with respect to t_1 is always the same, regardless of whether t_3 is enabled or not.**
- **Extensions to marking dependent rates have been defined.**

The definition of weights requires the identification of the sets of immediate transitions that can be simultaneously enabled in conflict.

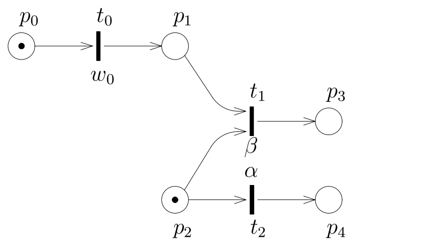
Such sets of transitions are called Extended Conflict Sets (ECSs).

When all the ECSs in a GSPN are known, the association of weights to transitions is easy, provided that no *confusion* exists.

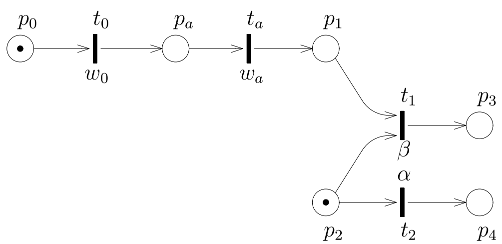
The structural and behavioural analysis of the marked PN with priority underlying the GSPN allows the qualitative study of the GSPN behaviour, and in particular the identification of

- **ECS**
- **confusion**

Confusion destroys the locality of conflicts

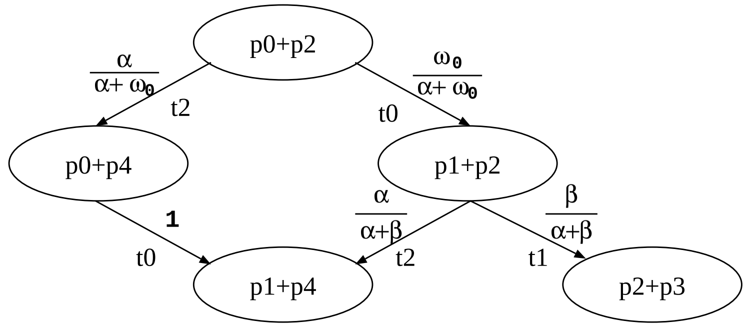


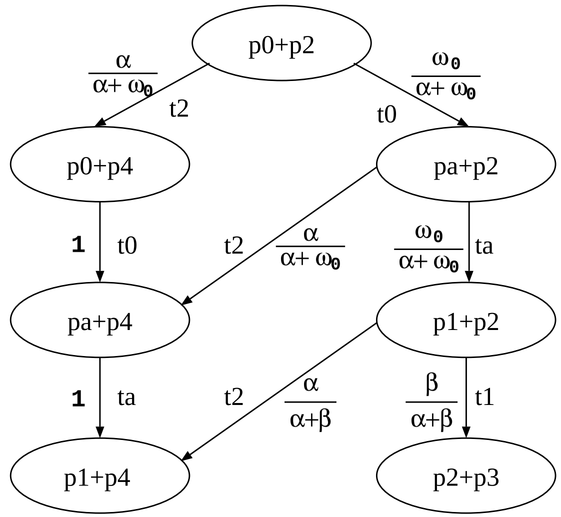
(a)



(b)

Confusion produces different transition probabilities





In the case of vanishing markings, the weights of the enabled n-immediate transitions can be used to determine which one will actually fire in a marking M that enables more than one conflicting n-immediate transitions.

When several transitions belonging to the same ECS are the only ones enabled in a given marking, one of them, say transition t_i, is selected as a candidate to fire with probability:

$$\mathbf{P}\{\ t_i \mid M\ \} \quad = \quad \frac{w_i}{W_I(M)}$$

where $W_I(M)$ is the weight of $ECS(t_i)$ in marking M, and is defined as follows:

$$W_I(M) \quad = \quad \sum_{k:\ t_k \in\ ECS(t_i) \cap E(M)} w_k$$

It may however happen that several ECSs comprising transitions of the same priority level are simultaneously enabled in a vanishing marking.

The characteristic of the subnets of n-immediate transitions of being confusion-free guarantees that the way in which this choice is performed is *irrelevant* with respect to the resulting stochastic model.

GSPNs can be proved to be isomorphic to Semi-Markov processes.

The analysis of a GSPN can be performed by studying a CTMC.

The state transition rate diagram of the MC corresponds to the *tangible* reachability graph of the GSPN.

The memoryless property of the exponential distribution makes the distinction among

- **resampling**
- **enabling memory**
- **age memory**

irrelevant.

The sojourn time in a tangible marking is exponentially distributed with a parameter that is the sum of the rates of all enabled timed transitions, so that the average time spent in marking M is given by:

$$E[SJ(M)] = \left[\sum_{t \in E(M)} W(t) \right]^{-1}$$

Numerical solution of GSPN models

An embedded Markov chain (EMC) can be recognized disregarding the concept of time and focusing the attention on the set of states of the semi-Markov process.

The specifications of a GSPN system are sufficient for the computation of the transition probabilities of such a chain.

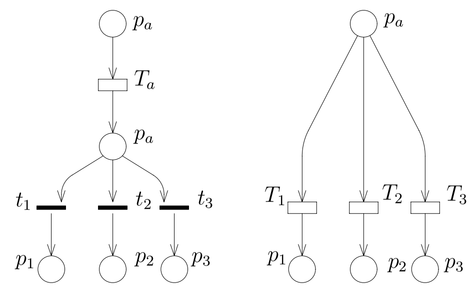
Several techniques have been devised for restricting the computation to reduced models accounting for the tangible markings only.

Three different approaches can be used:

- Identify a reduced Embedded Markov Chain defined over the set of tangible markings only;

- Compute the transition probabilities among tangible markings directly by applying (on-the-fly) a depth-first algorithm that explores all complete vanishing paths emanating from each tangible state.
  The method assumes that no loops among vanishing states exist and memory saving is traded-off with (possible) repeated computations;

- Reduce the GSPN to an equivalent SPN obtained by fusing immediate transitions with preceding timed transitions using an algorithm that in the simple cases produces the following reductions



Computational considerations

The mathematically elegant solution of the model using the REMC suffers in practice of the difficulties deriving from the size of the CTMC and from time-scale differences that may exist among the firing rates of the transitions of a model.

Approaches that can be used to overcome these difficulties are the following:

- Transient solution
  - Uniformization method
- Steady state solution
  - Time scale decomposition
  - Tensor algebras and compositionality
  - Symmetries and exact lumping
- Simulation

Performance indices

From the steady-state probability distribution of markings it is possible to obtain several performance indices that are the basis for a quantitative evaluation of the behaviour of the GSPN.

As in the case of SPNs, these results can be computed using the unifying approach based on the definition of reward functions.

Tools

The applicability of the [(G)S]PN approach to anything but the smallest toy examples rests on the availability of efficient tools for the

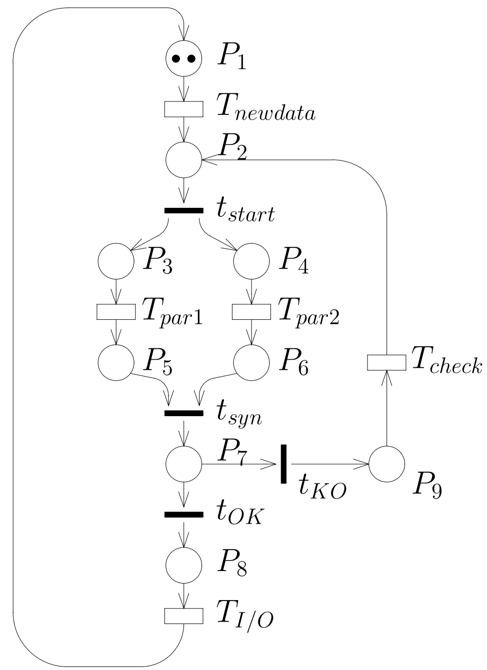
- **model construction (*top-down, bottom-up, compositionality*)**
- **model debugging (*structural analysis*)**
- **definition of performance indices**
- **model solution (*analysis and/or simulation*)**
- **computation of aggregate results**
- **display of results**

Good software tools are a must.

The user-friendliness and the graphical capabilities of the tool are of paramount importance.

Example

The GSPN description of a simple parallel system



transition	rate	value	semantics
$T_{newdata}$	λ	**1**	**infinite-server**
T_{par1}	μ_1	**10**	**single-server**
T_{par2}	μ_2	**5**	**single-server**
$T_{I/O}$	ν	**25**	**single-server**
T_{check}	θ	**0.5**	**single-server**

transition	weight	priority	ECS
t_{start}	**1**	**1**	**1**
t_{syn}	**1**	**1**	**2**
t_{OK}	**99**	**1**	**3**
t_{KO}	**1**	**1**	**3**

The consistency check operation results in a success 99% of the times, and in a failure 1% of the times.

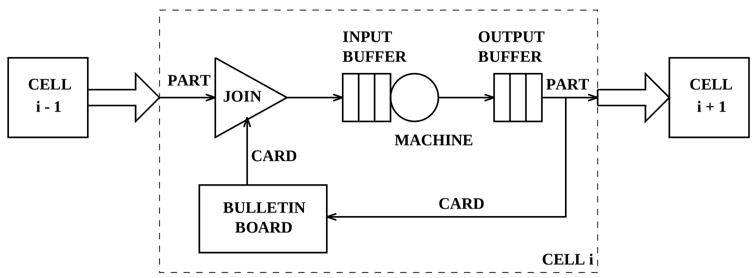
The GSPN model generates

- **20 tangible markings**
- **18 vanishing markings**

As an example, with the numerical values chosen for the model parameters, the probability of at least one process waiting for synchronization is computed to be 0.238.

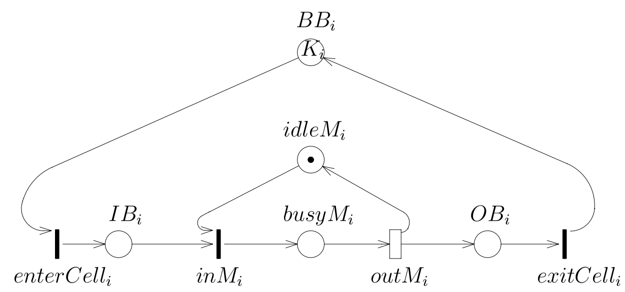
A case study

a kanban system



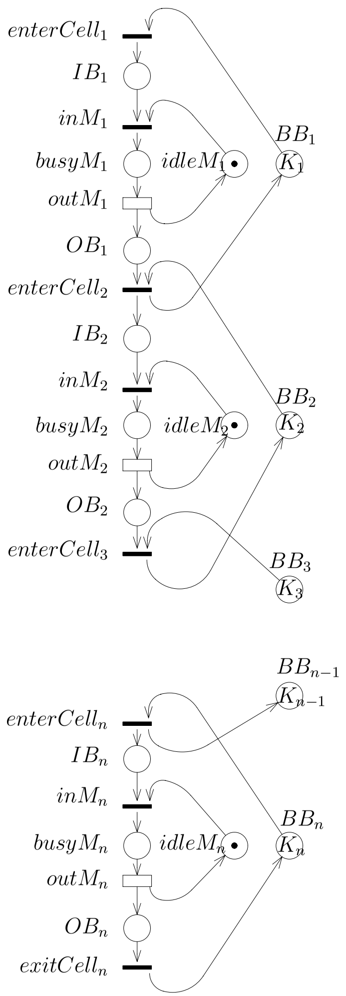
A kanban cell and the parts and cards that flow into and out of it

The basic model



GSPN model of a single Kanban cell

n-cell sequential Kanban



Qualitative analysis

A n-cell Kanban model has $2n$ minimal P-semiflows, whose associated P-invariants are:

$\forall i, 1 \leq i \leq n$:

$$M(BB_i) + M(IB_i) + M(busyM_i) + M(OB_i) = K_i$$

$$M(idleM_i) + M(busyM_i) = 1$$

It follows that:

- **The number of parts in cell i is at most K_i, the number of cards in the cell;**
- **Each machine can process only one part at a time;**
- **Places $idleM_i$ and $busyM_i$ are mutually exclusive.**

All transitions of the GSPN model are covered by a *single* minimal T-semiflow: it represents the deterministic flow of the unique type of parts processed by the system.

The net behaviour is deterministic: no structural conflicts exist, hence neither effective conflicts nor confusion can ever arise.

Quantitative analysis

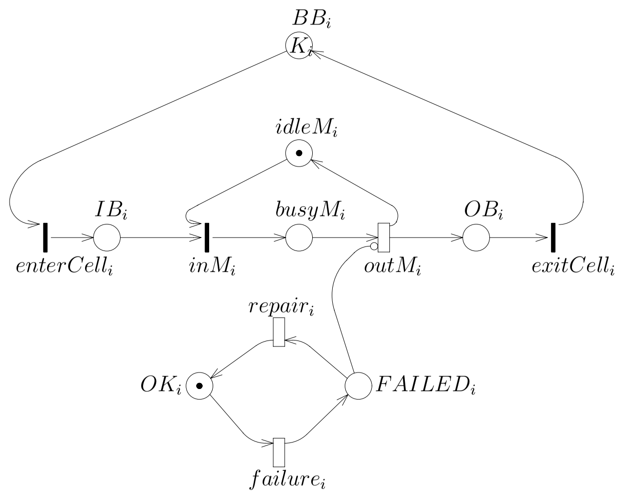
We consider K cards and $n = 5$ cells of equal machine time (the rate of transitions $outM_i$ is 4.0)

First case: Input and output inventory in the cells

	Input buffer inventory			Output buffer inventory		
Cell	1 Card	2 Cards	3 Cards	1 Card	2 Cards	3 Cards
1	0.486	1.041	1.474	0.514	0.958	1.526
2	0.486	1.040	1.470	0.383	0.713	1.131
3	0.486	1.047	1.478	0.282	0.524	0.811
4	0.486	1.056	1.490	0.170	0.316	0.472
5	0.486	1.073	1.515	0.000	0.000	0.000

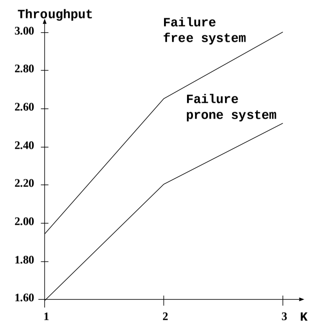
The input inventory is fairly constant, while the output inventory decreases as the cell position increases.

Second case: fault free versus failure prone systems

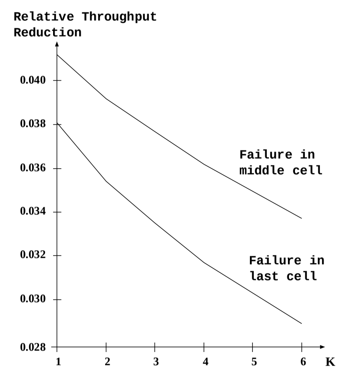


Model of a Kanban cell that can fail

- **Cells can fail independently;**
- **Failure rate is 0.02;**
- **Repair rate is 0.4.**



Failure free vs. "all cells can fail"



Middle cell failure vs. final cell failure

In a perfectly balanced Kanban system the cell performance is position-dependent.

CONCLUSIONS

Stochastic Petri net techniques are attractive because they provide a performance evaluation approach based on a formal description.

This allows the use of the same language for the

- **specification**
- **validation**
- **performance evaluation**
- **implementation**
- **documentation**

of a system.

Two are the main directions of the research being presently conducted in the field of GSPN-based performance evaluation.

1. **Extensions of the GSPN analysis approach to environments in which tokens possess an identity has already been proposed by several authors, and more work is being performed to obtain an environment with a high descriptive power in which the model specification is simple.**

2. **Various approaches are being pursued for the reduction of the complexity of the solution computation with stochastic techniques, possibly producing only partial or approximate results.**

Successes in these two fields would make GSPN a prominent modeling technique in the whole area of distributed systems.