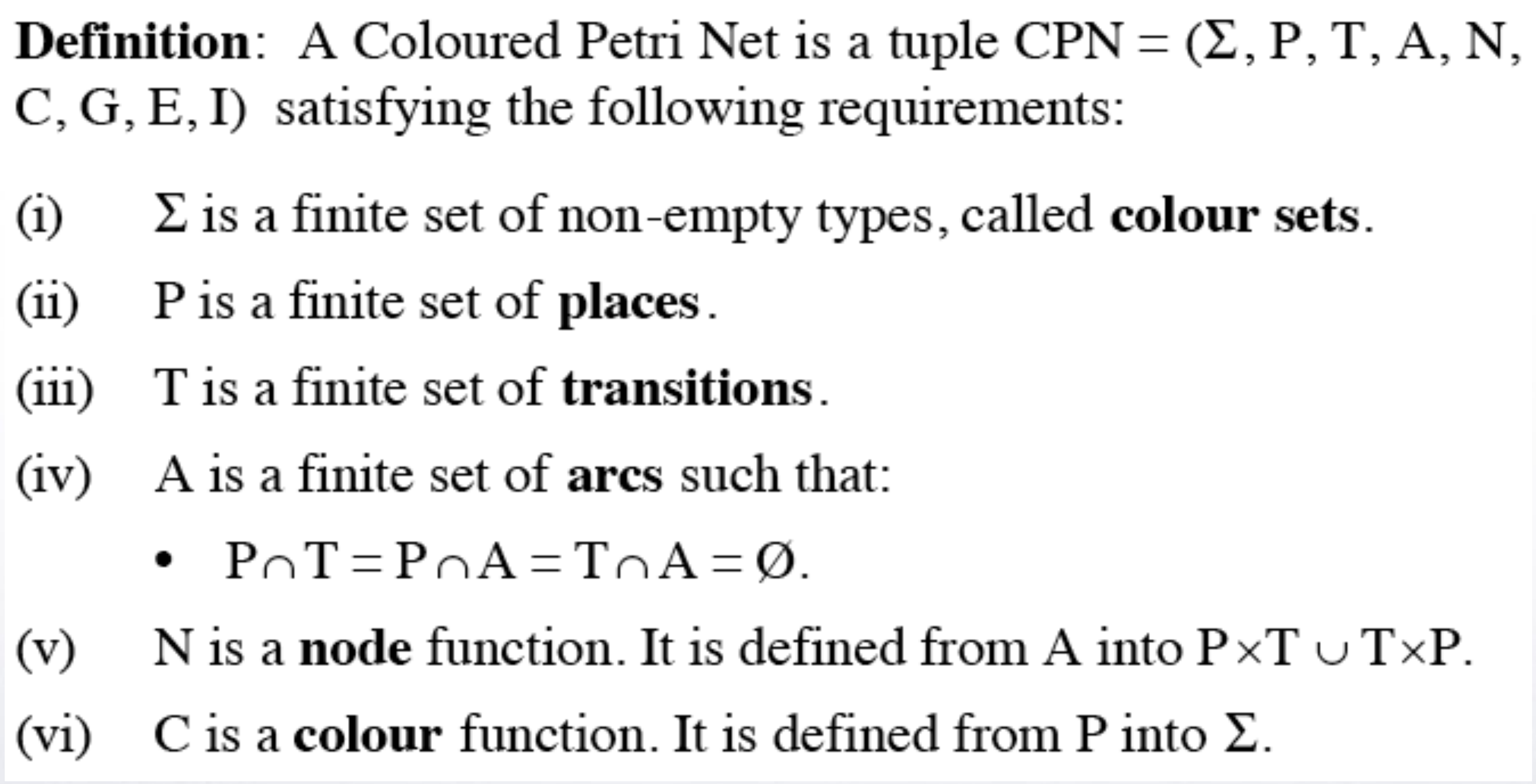
PMR 5237
Modelagem e Design de Sistemas Discretos em Redes de Petri

Aula 9 : Análise de Propriedades e Técnicas de Modelagem

Prof. José Reinaldo Silva
reinaldo@usp.br

CPN : definição formal



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**.

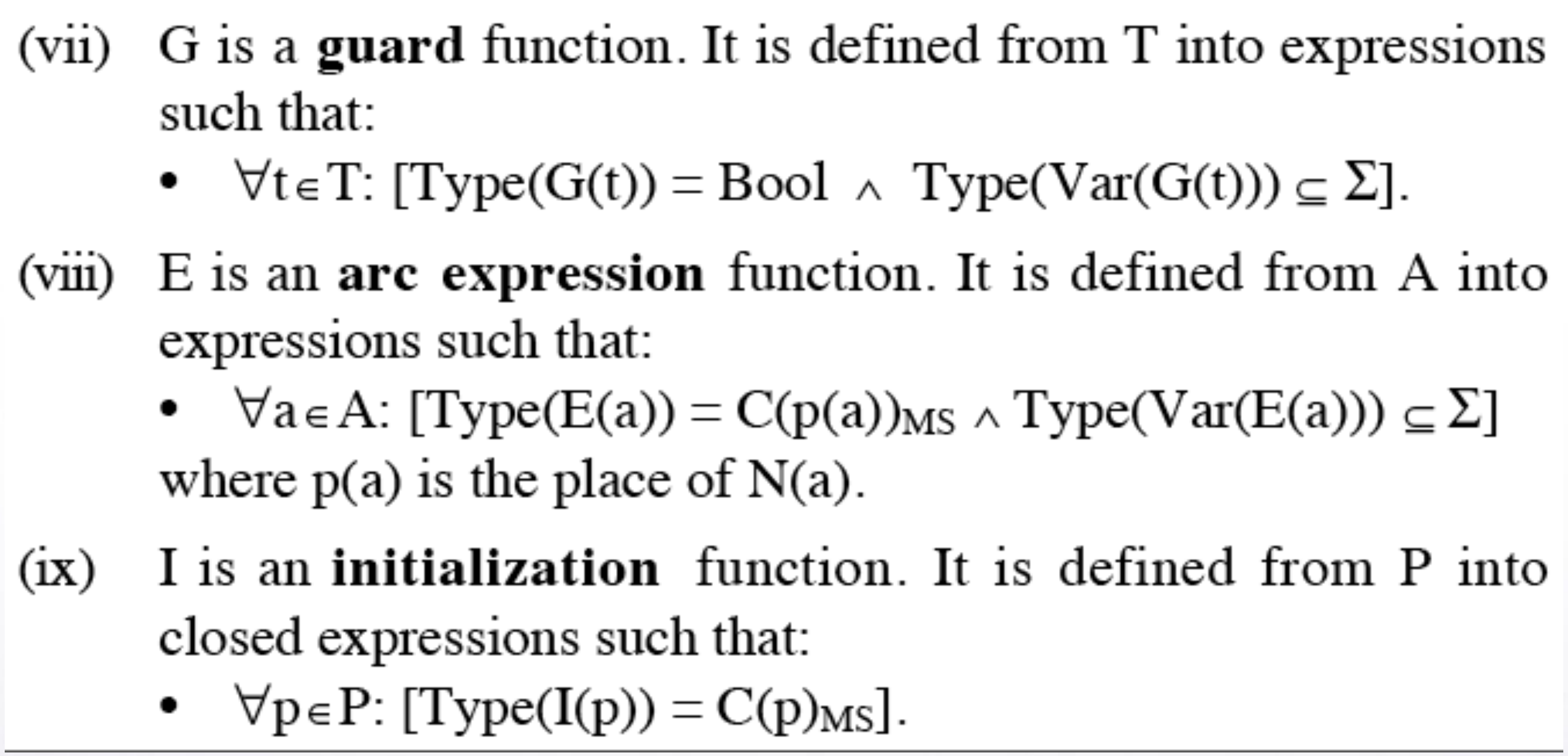
(iii) T is a finite set of **transitions**.

(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:

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

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

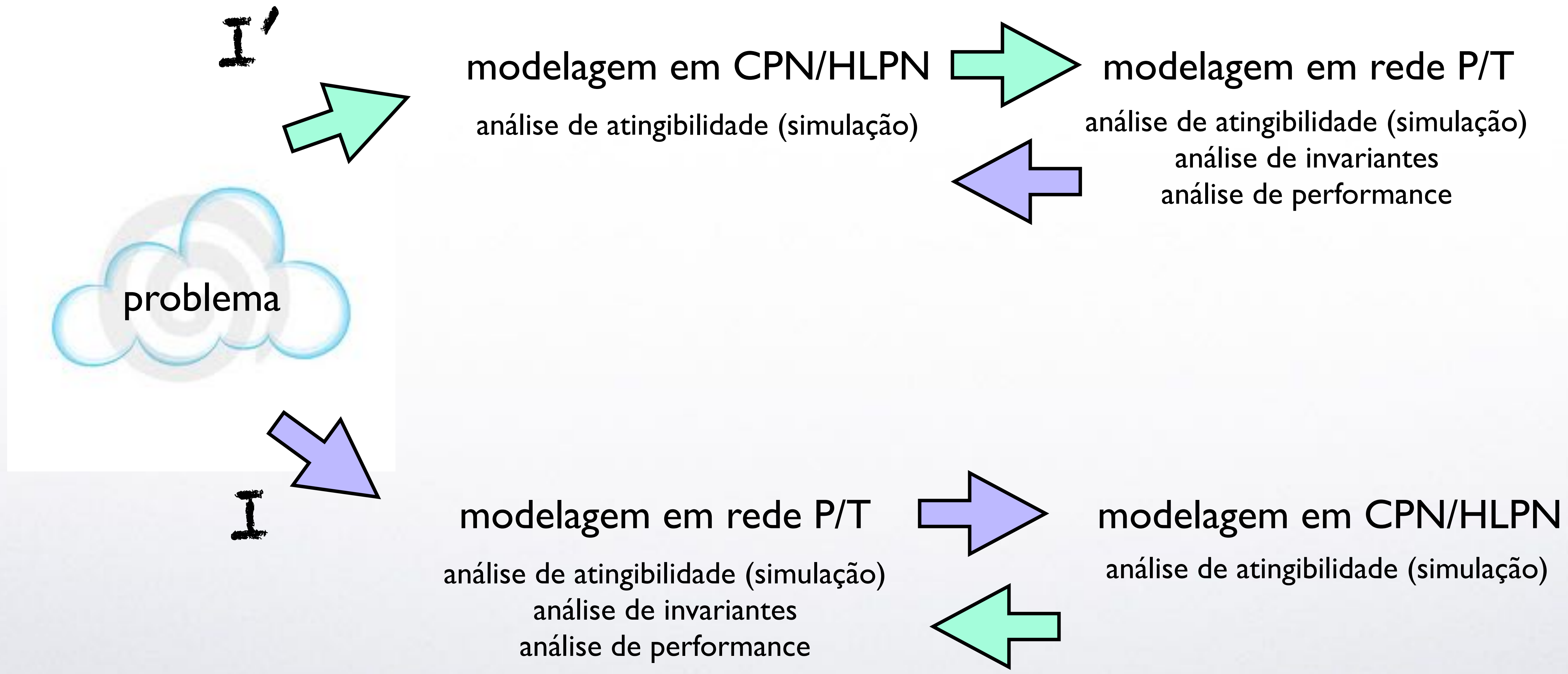
- $\forall a \in A$: $[\mathrm{Type}(E(a)) = C(p(a))_{MS} \wedge \mathrm{Type}(\mathrm{Var}(E(a))) \subseteq \Sigma]$

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:

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

Modelagem e análise em Redes Colorida/Alto Nível



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.

Correct interpretation means that the "behavior" of practical system (described in some language) is a metaphor for the high level net (syntactically and semantically).

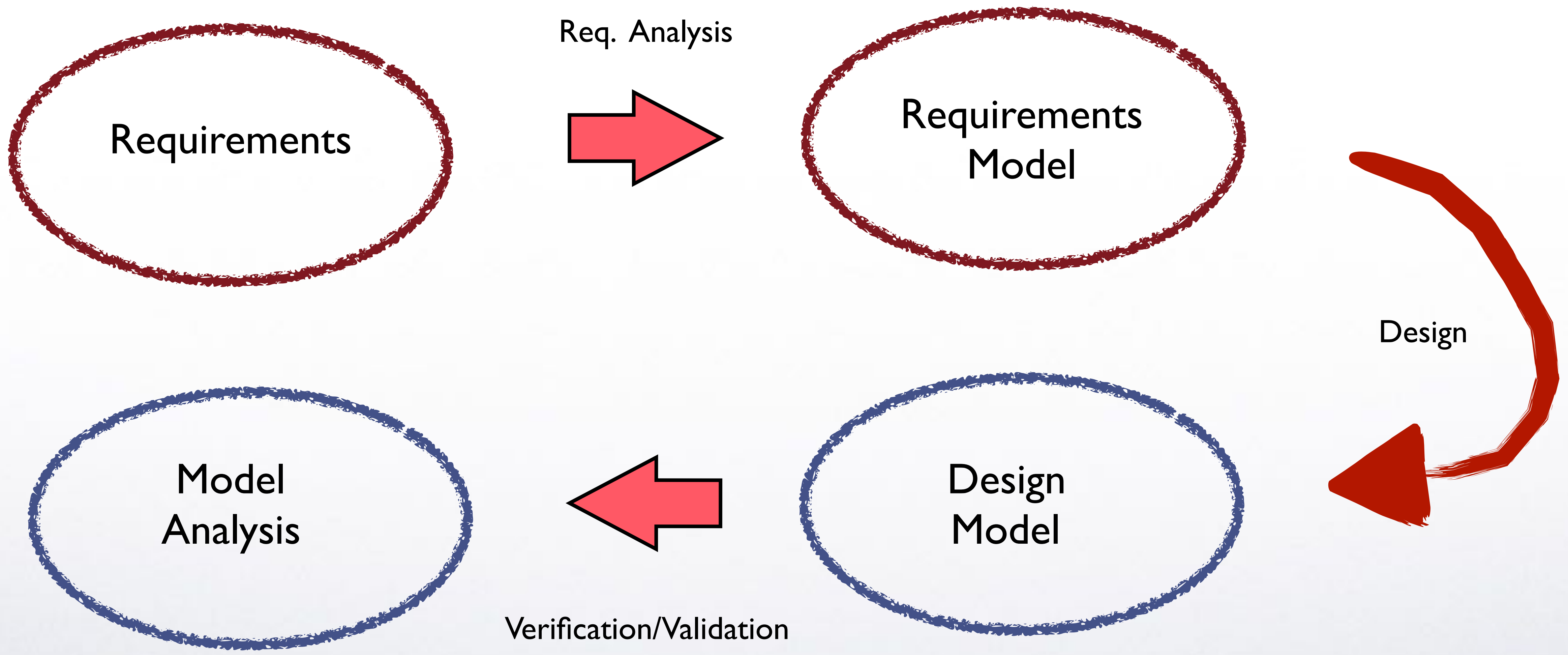
Even if environments can be an apprentice for that there is no guarantee to get a good metaphor. Only going through the modeling, analysis and verification we can be certain about "correctness".

Interpretations can also be :

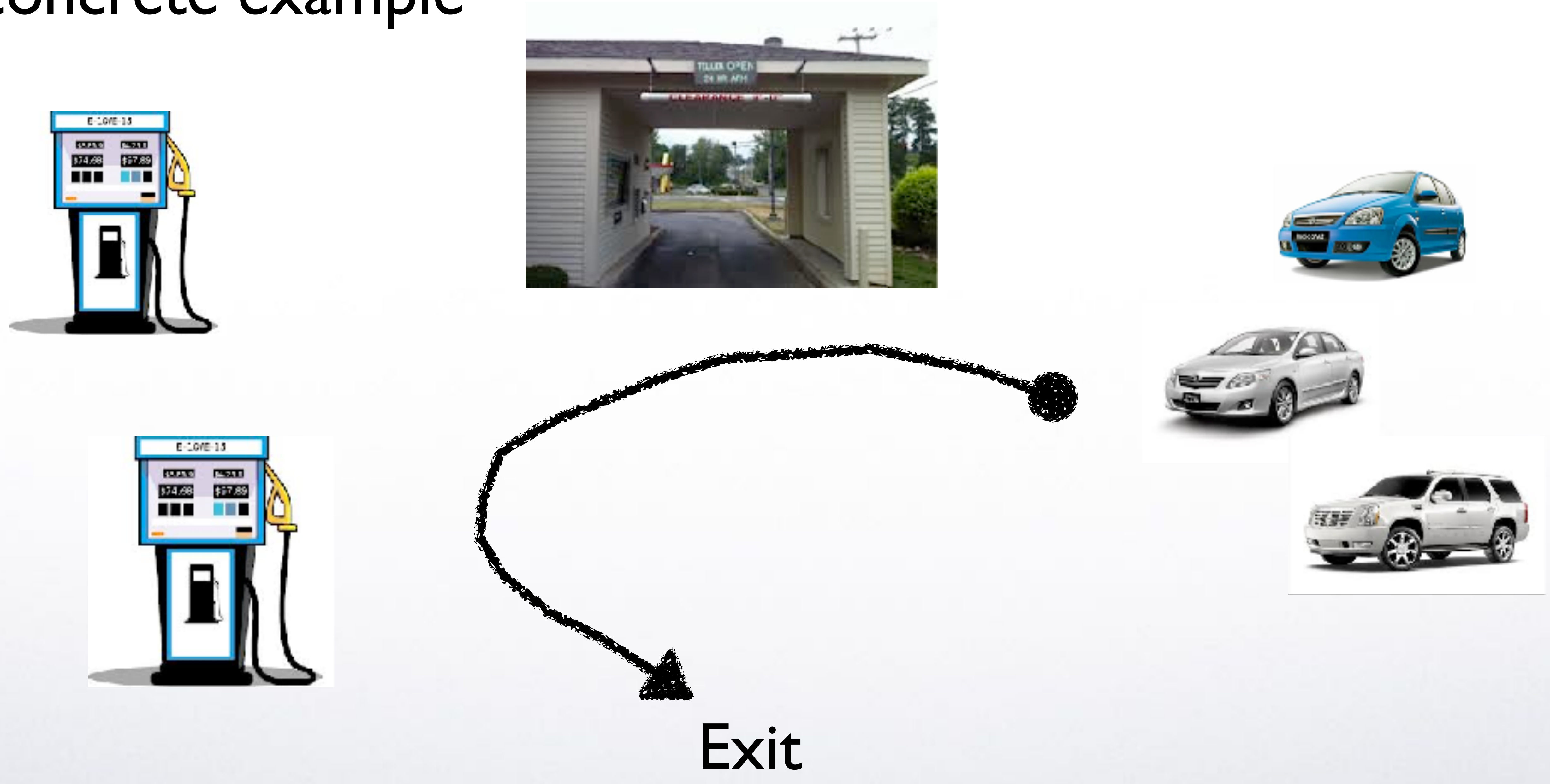
abstract (like in requirements analysis)

concrete (like in PLC programming)

Use of Petri Nets in Design

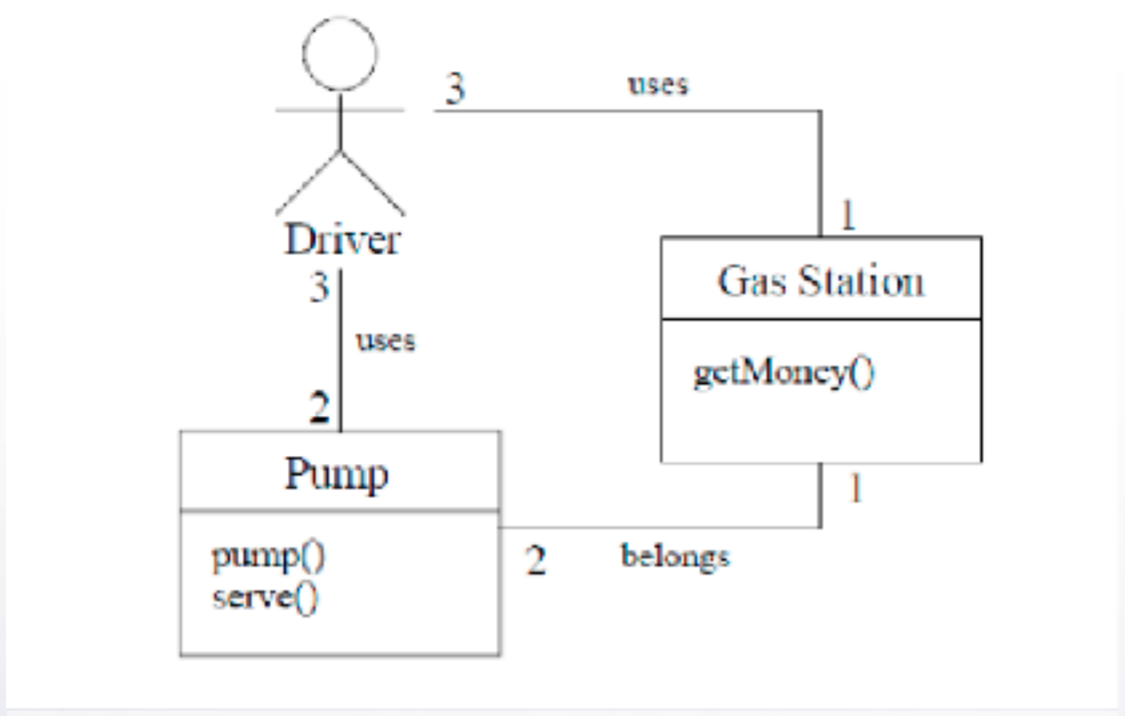


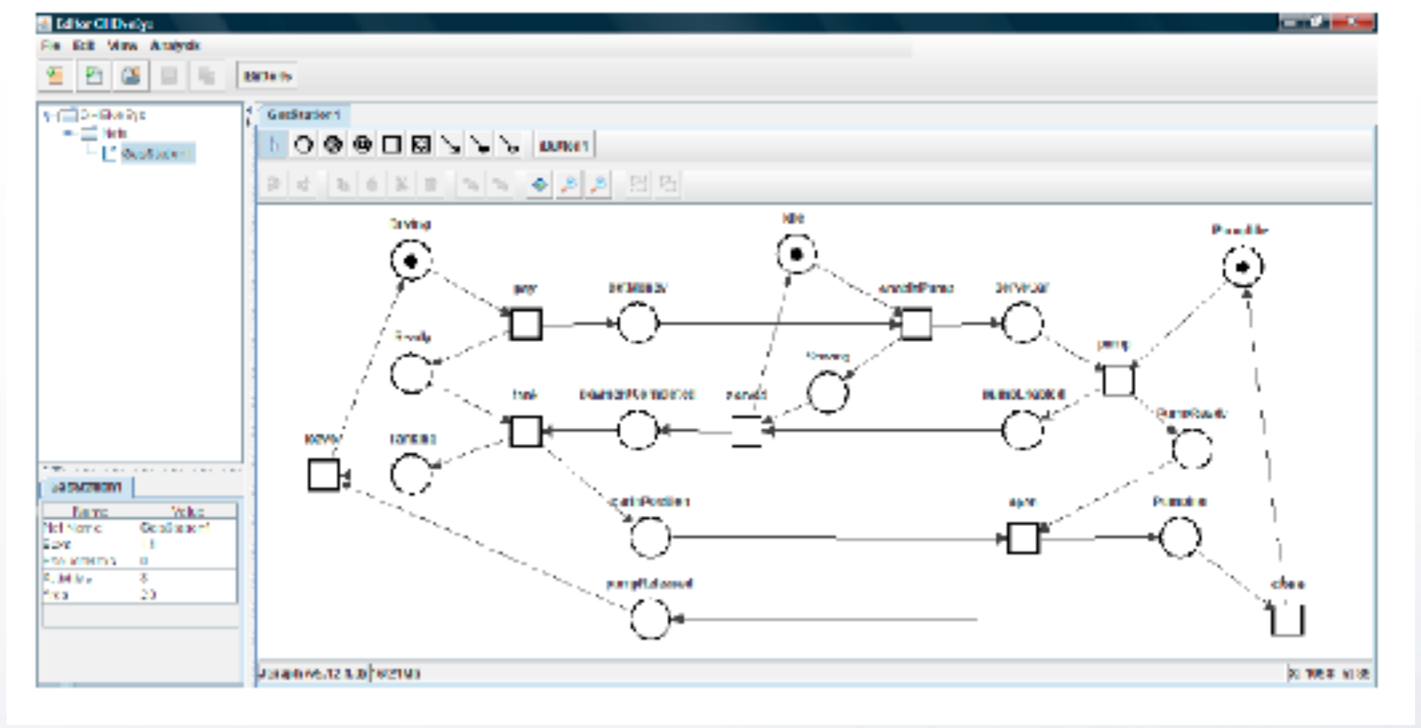
A concrete example



The Gas Station Problem

del Foyo, P.M.G., Salmon, A.O., Silva, J.R.; Requirements Analysis of Automated Problems Using UML/Petri Nets, Proc. of the 21st. Congress of Mech. Eng., Natal, 2011.

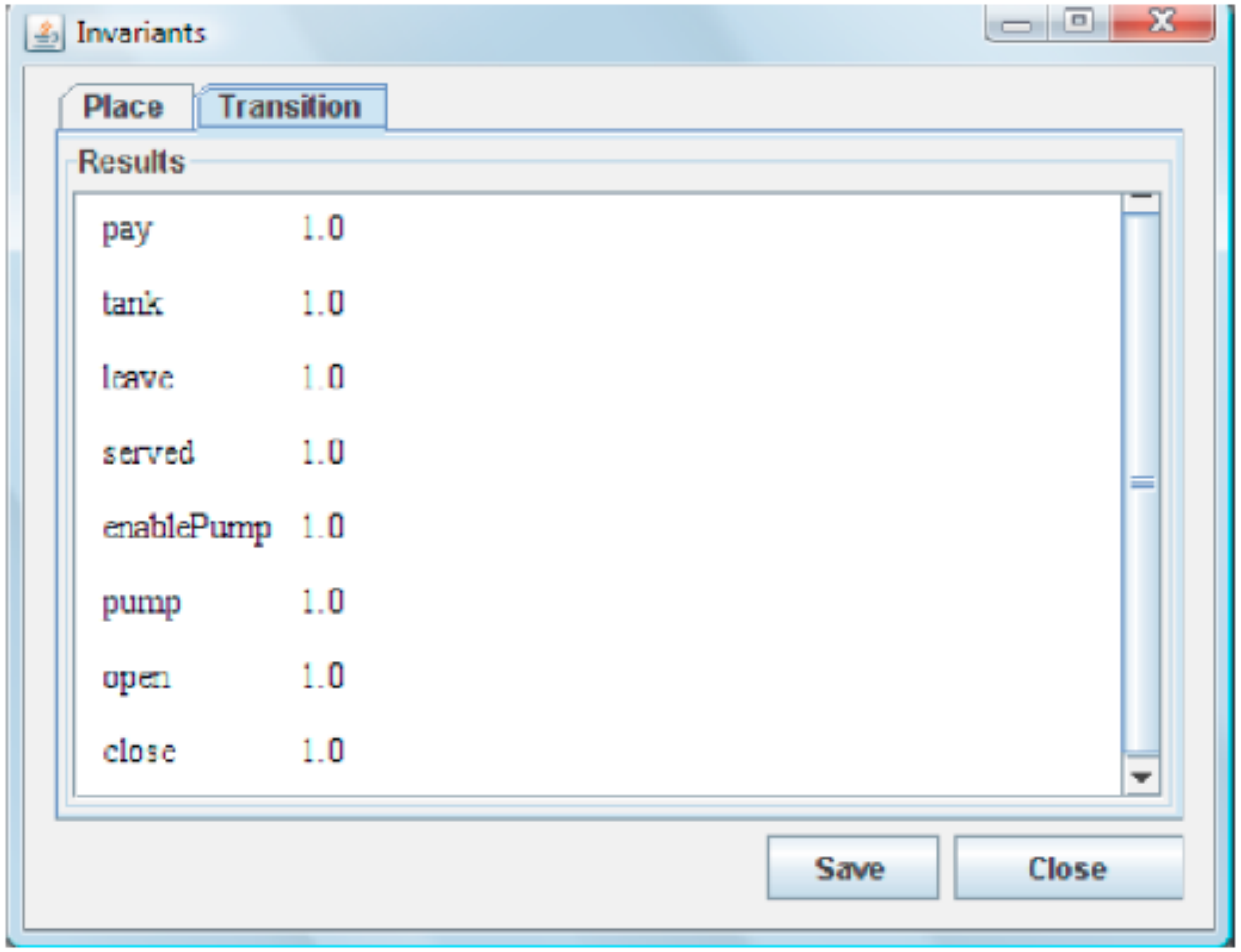




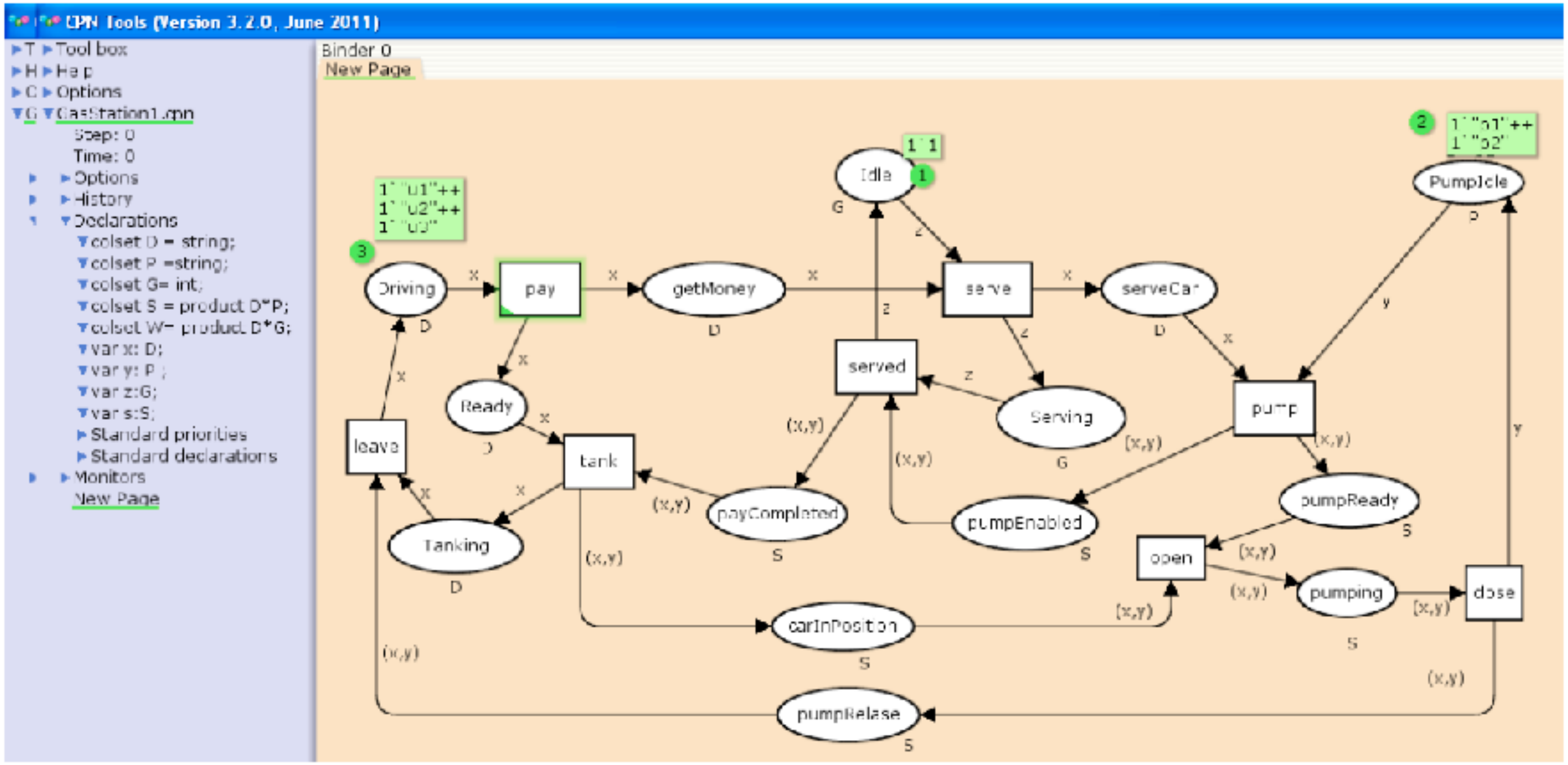
Baresi, L. and Pezze, M., 2001. "Improving UML with petri nets". In *UNIGRA 2001, Uniform Approaches to Graphical Process Specification Techniques (a Satellite Event of ETAPS 2001)*. Elsevier, Vol. 44 of *Electronic Notes in Theoretical Computer Science*, pp. 107–119.

Requirements Analysis

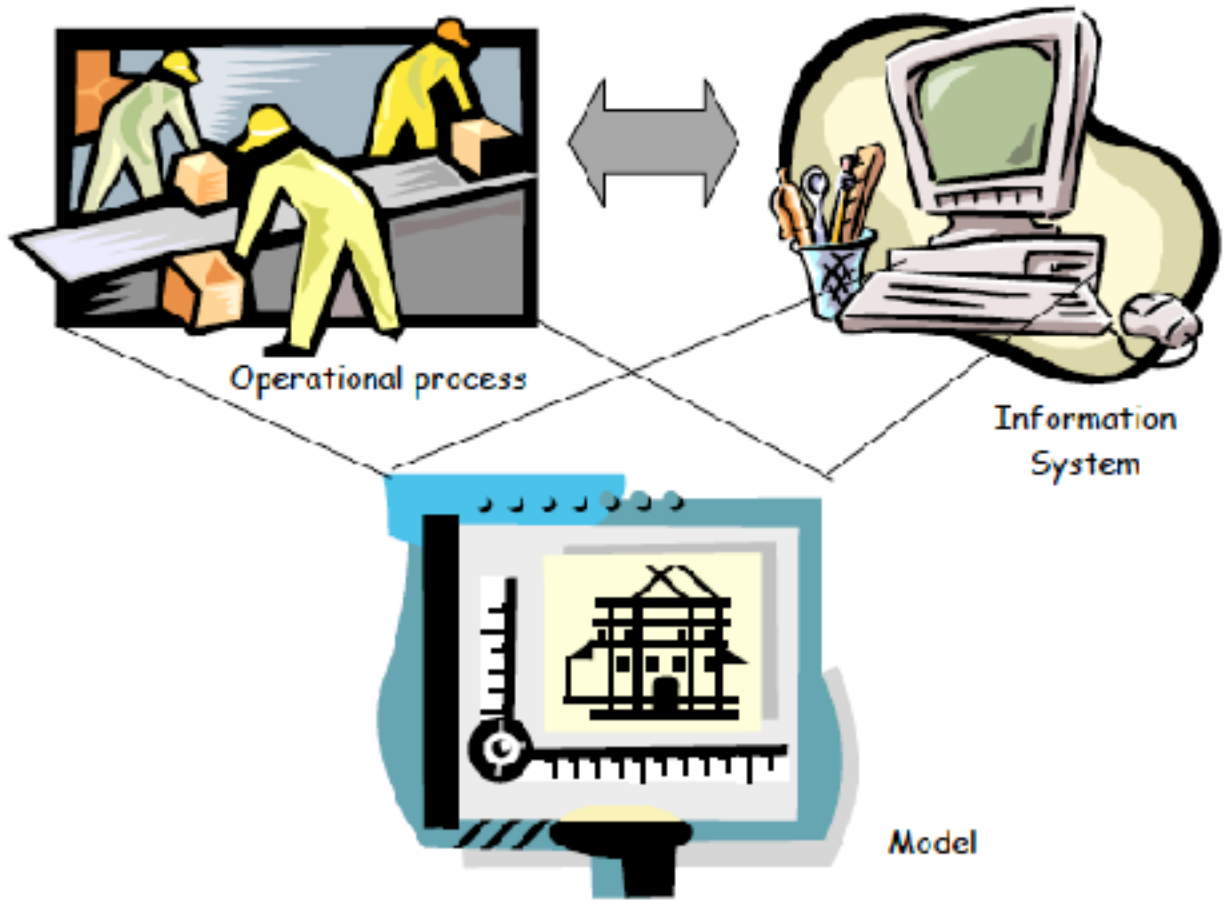
All transitions must fire to get a complete cycle for using the gas station (it does not matter how many Pumps it has)



pay	1.0
tank	1.0
leave	1.0
served	1.0
enablePump	1.0
pump	1.0
open	1.0
close	1.0


CPN Tools (Version 3.2.0, June 2011)

```
▸T ▸Tool box
▸H ▸Help
▸C ▸Options
▾G ▾GasStation1.cpn
      Step: 0
      Time: 0
   ▸ ▸Options
   ▸ ▸History
   ▾ ▾Declarations
        ▾colset D = string;
        ▾colset P =string;
        ▾colset G= int;
        ▾colset S = product D*P;
        ▾colset W= product D*G;
        ▾var x: D;
        ▾var y: P ;
        ▾var z:G;
        ▾var s:S;
        ▸Standard priorities
        ▸Standard declarations
   ▸ ▸Monitors
      New Page
```



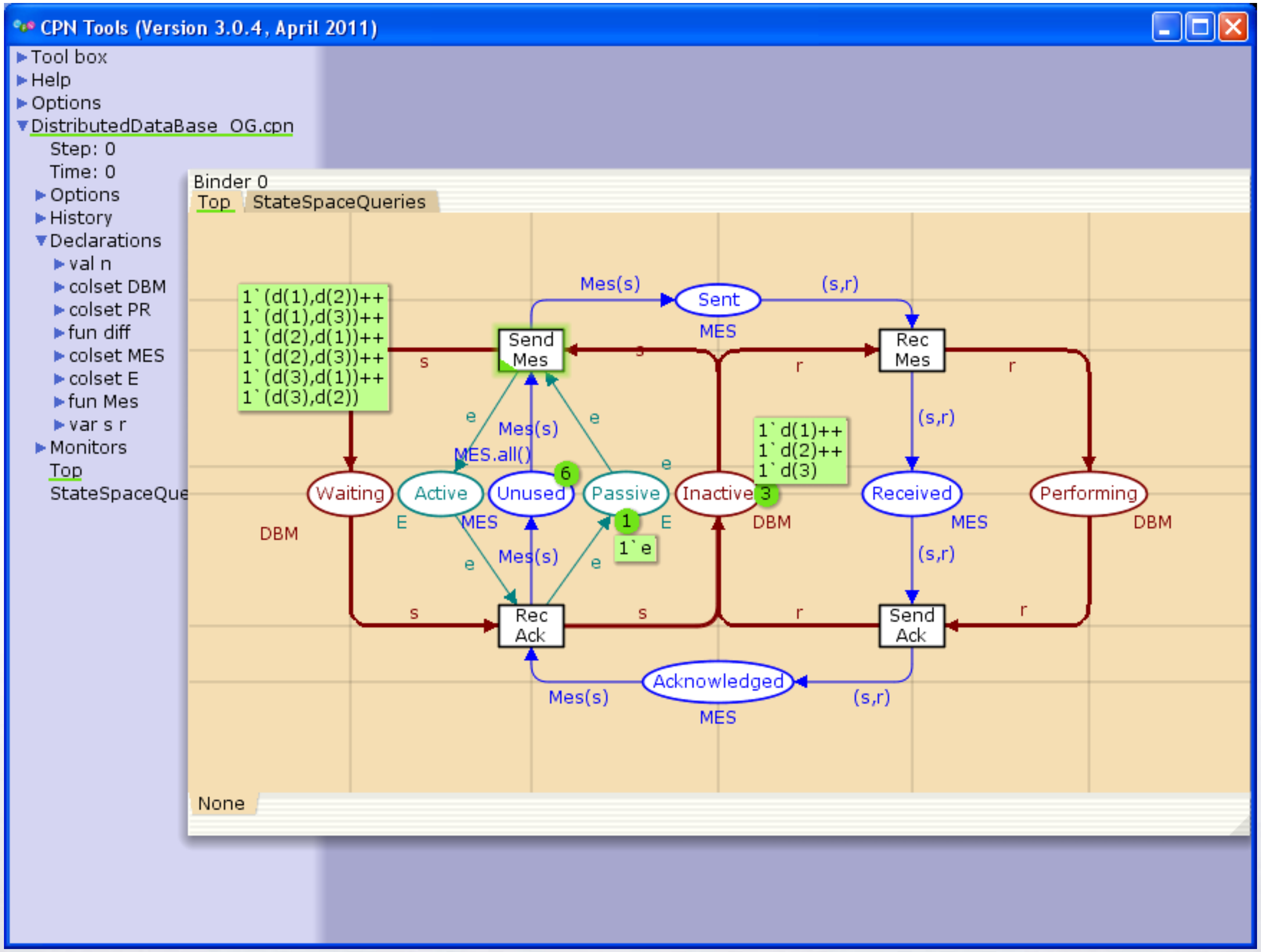
- Analysis is typically model-driven to allow e.g. what-if questions.
- Models of both operational processes and/or the information systems can be analyzed.
- Types of analysis:
  - *validation*
  - *verification*
  - *performance analysis*

1. Reachability/coverability graph
2. Structural techniques
   - Place and transition invariants
   - Marking equation
   - Traps, siphons, etc.
3. Simulation

- Each can be applied to both classical and high-level Petri nets.
- Nevertheless, for the second we restrict ourselves to classical Petri nets.

Mapping technique/use:

- reachability graph (validation, verification)
- invariants (validation, verification)
- simulation (validation, performance analysis)


CPN Tools (Version 3.0.4, April 2011)
Tool box
Help
Options
DistributedDataBase_OG.cpn
Step: 0
Time: 0
Options
History
Declarations
val n
colset DBM
colset PR
fun diff
colset MES
colset E
fun Mes
var s r
Monitors
Top
StateSpaceQue
Binder 0
Top
StateSpaceQueries
1`(d(1),d(2))++
1`(d(1),d(3))++
1`(d(2),d(1))++
1`(d(2),d(3))++
1`(d(3),d(1))++
1`(d(3),d(2))
Mes(s)
Sent
MES
(s,r)
Send Mes
Rec Mes
s
s
r
r
e
Mes(s)
MES.all()
e
e
6
1`d(1)++
1`d(2)++
1`d(3)
(s,r)
Waiting
Active
Unused
Passive
Inactive
3
Received
Performing
DBM
E
MES
1
E
DBM
MES
DBM
1`e
e
Mes(s)
e
(s,r)
s
s
r
r
Rec Ack
Send Ack
Acknowledged
Mes(s)
(s,r)
MES
None

CPN Analysis

CPN Analysis follow the same procedures than the classical analysis and face the same problems, even if has new formal resources to include.

Again, we have a state/transition approach, in a discrete flavor, with the possibility of explosion of the number of combinations of states, that is, in the composition of processes.

Modelagem clássica

As redes possuem propriedades típicas dos esquemas que as tornam
Uma excelente representação formal para sistemas (dinâmicos) discretos,
Entre os quais figuram :

o princípio da dualidade
o princípio da localidade
o princípio da concorrência
o principio da representação gráfica
o princípio da representação algébrica

Special configurations

As in the classic net systems, we detached specific net configurations that constitute a challenge in the analysis process (or situations to be avoided, in order to have the desirable system). Some of this situations are

- Conflict
- Branching and synchronization
- Deadlocks

Distribution and concurrency

As before, everything is based on the concept of locality. According to that, individual states could be classified as independent, and, in such a case, they could be grouped in macro states called cases. Conversely, independent transitions could be also grouped in steps.

Thus, cases and steps could be arranged in a dual way, generating a more abstract net.

Classic and HL modeling

Concluding, the modeling in Classic P/T nets (taken as the abstract archetype of the classic nets) is very similar to the modeling in CPN (or in any high level net).

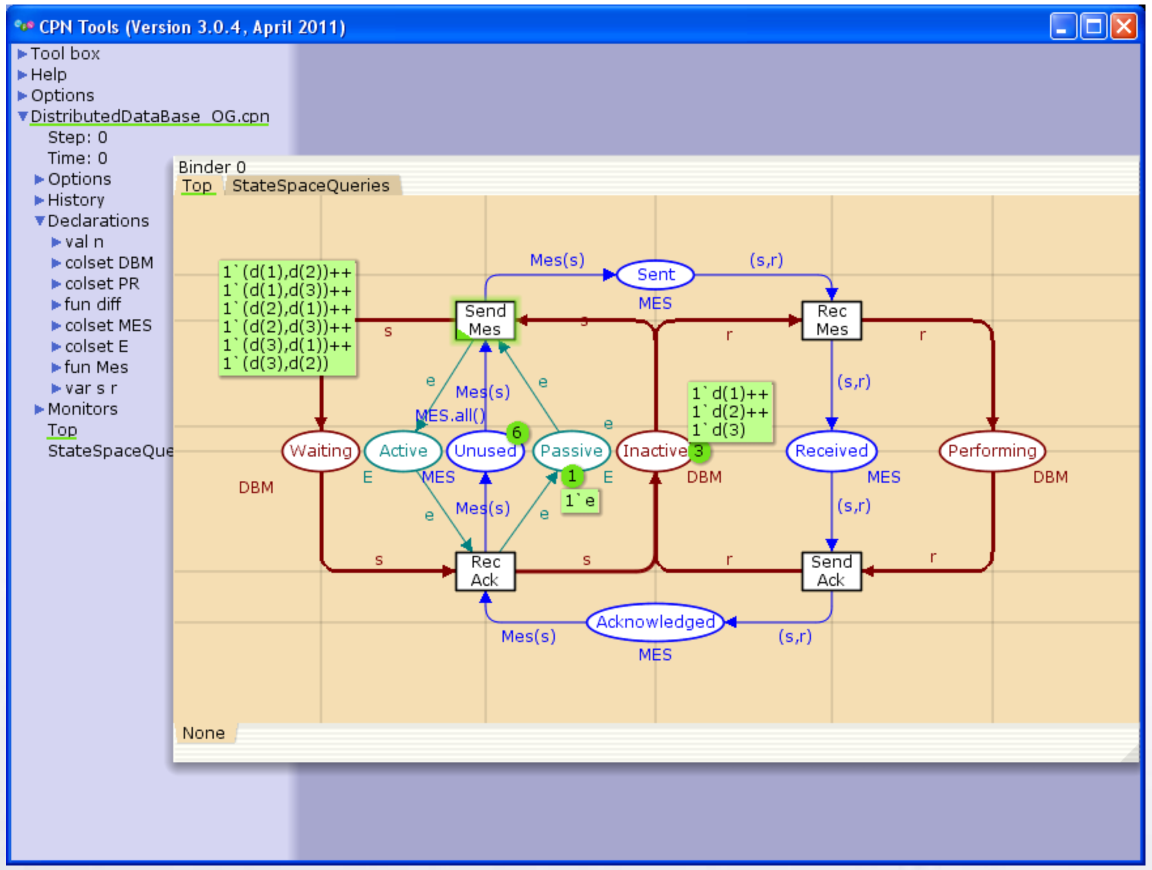
On the other hand, the inclusion of type theory, or even a simple distinction in the tokens should not be underestimated as good resources to analyze complex systems more comfortably. Such advances is what open new applications to PN and what could make the modeling, analysis (and more recently the verification process) reliable.

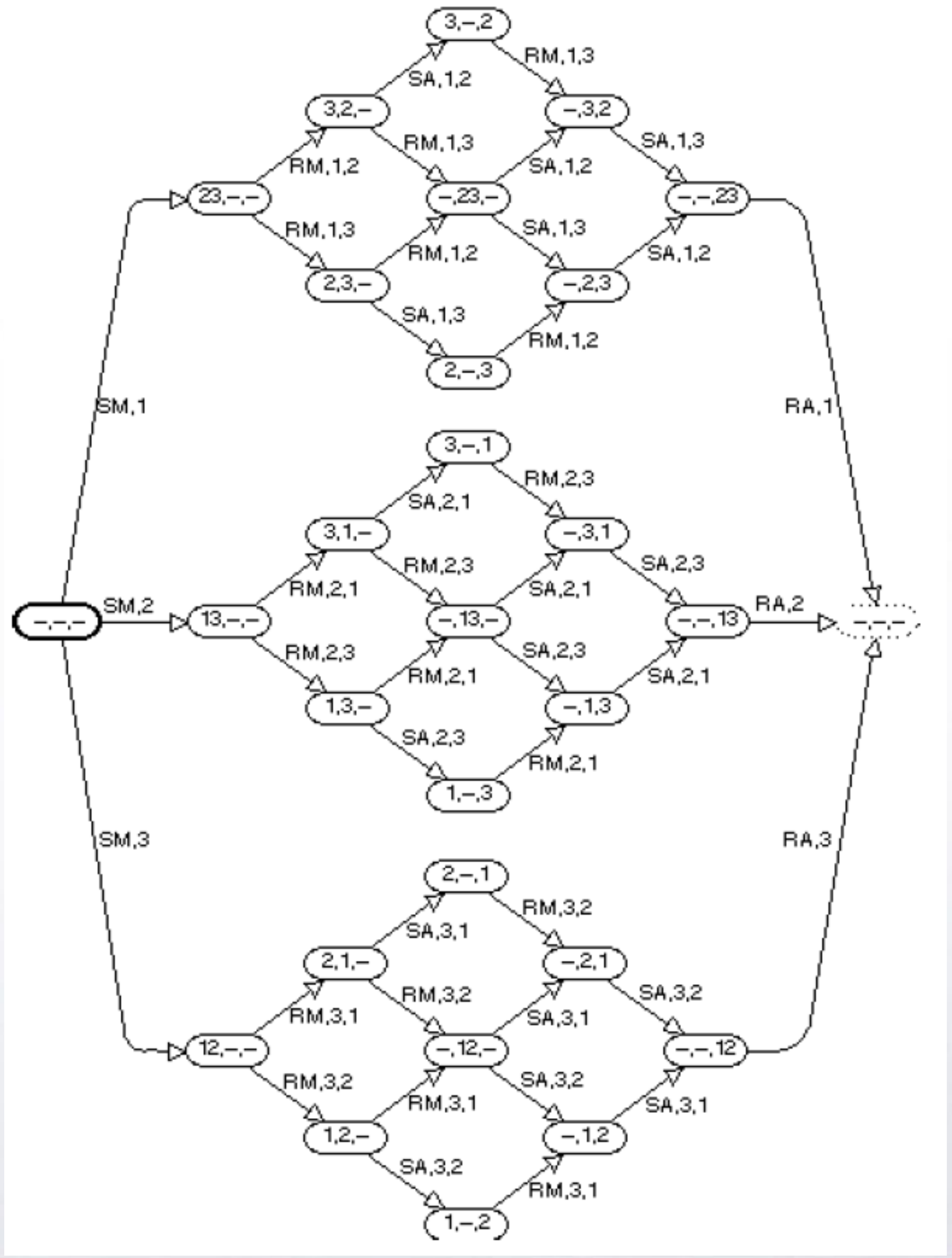
Extensions

Even extension elements are used in a similar way. So far we have seen two very important extension elements:

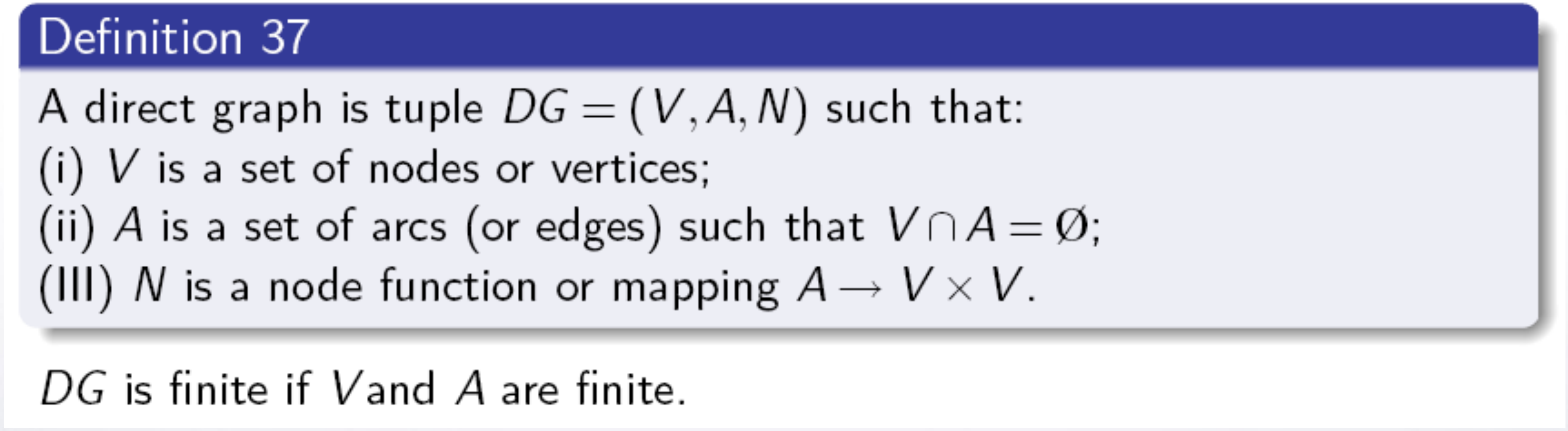
- Gates
- Hierarchy

Analysis in CPN Nets





Directed Graphs

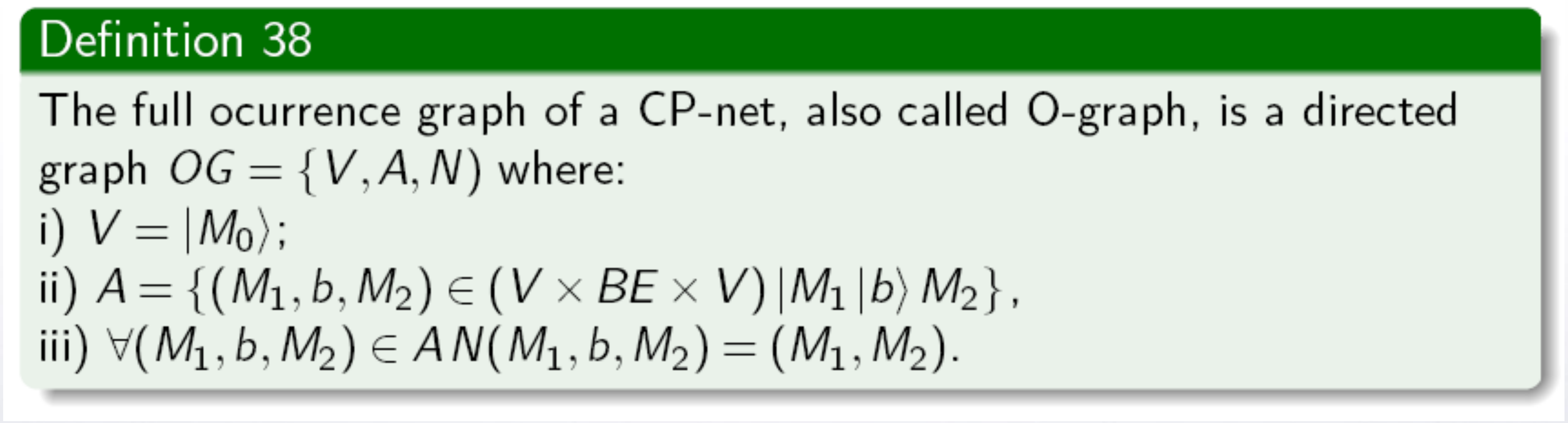


Definition 37

A direct graph is tuple $DG = (V, A, N)$ such that:
(i) V is a set of nodes or vertices;
(ii) A is a set of arcs (or edges) such that $V \cap A = \emptyset$;
(III) N is a node function or mapping $A \rightarrow V \times V$.

DG is finite if $V$and A are finite.

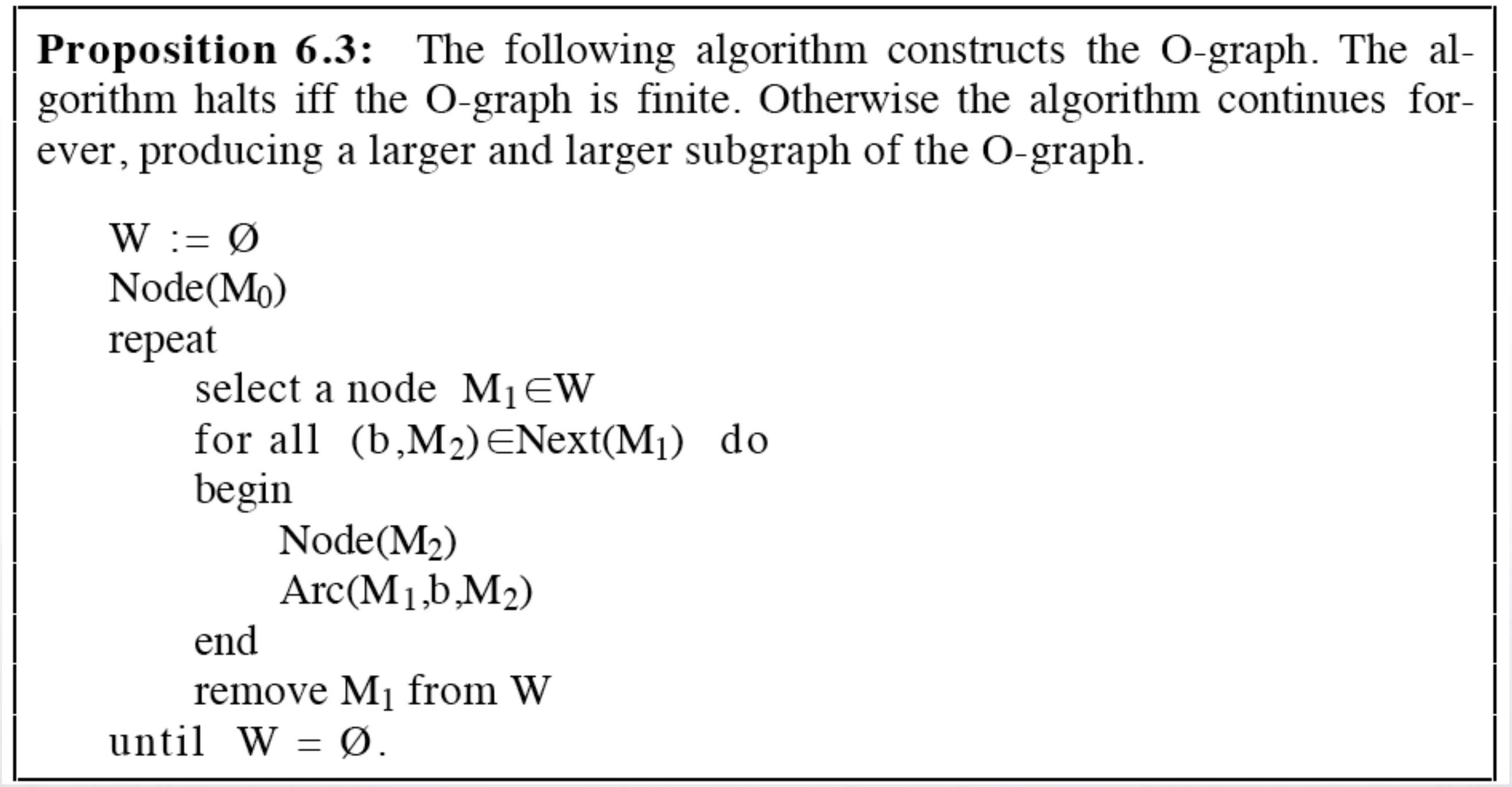
O-Graph



Definition 38

The full ocurrence graph of a CP-net, also called O-graph, is a directed graph $OG = \{V, A, N)$ where:
i) $V = |M_0\rangle$;
ii) $A = \{(M_1, b, M_2) \in (V \times BE \times V) | M_1 |b\rangle M_2\}$,
iii) $\forall (M_1, b, M_2) \in A\, N(M_1, b, M_2) = (M_1, M_2)$.

O Graph algorithm



Proposition 6.3: The following algorithm constructs the O-graph. The algorithm halts iff the O-graph is finite. Otherwise the algorithm continues forever, producing a larger and larger subgraph of the O-graph.

```
W := Ø
Node($M_0$)
repeat
      select a node $M_1 \in W$
      for all $(b, M_2) \in Next(M_1)$ do
      begin
            Node($M_2$)
            Arc($M_1$,b,$M_2$)
      end
      remove $M_1$ from W
until W = Ø.
```

The invariant method

We first *construct* a set of place invariants.

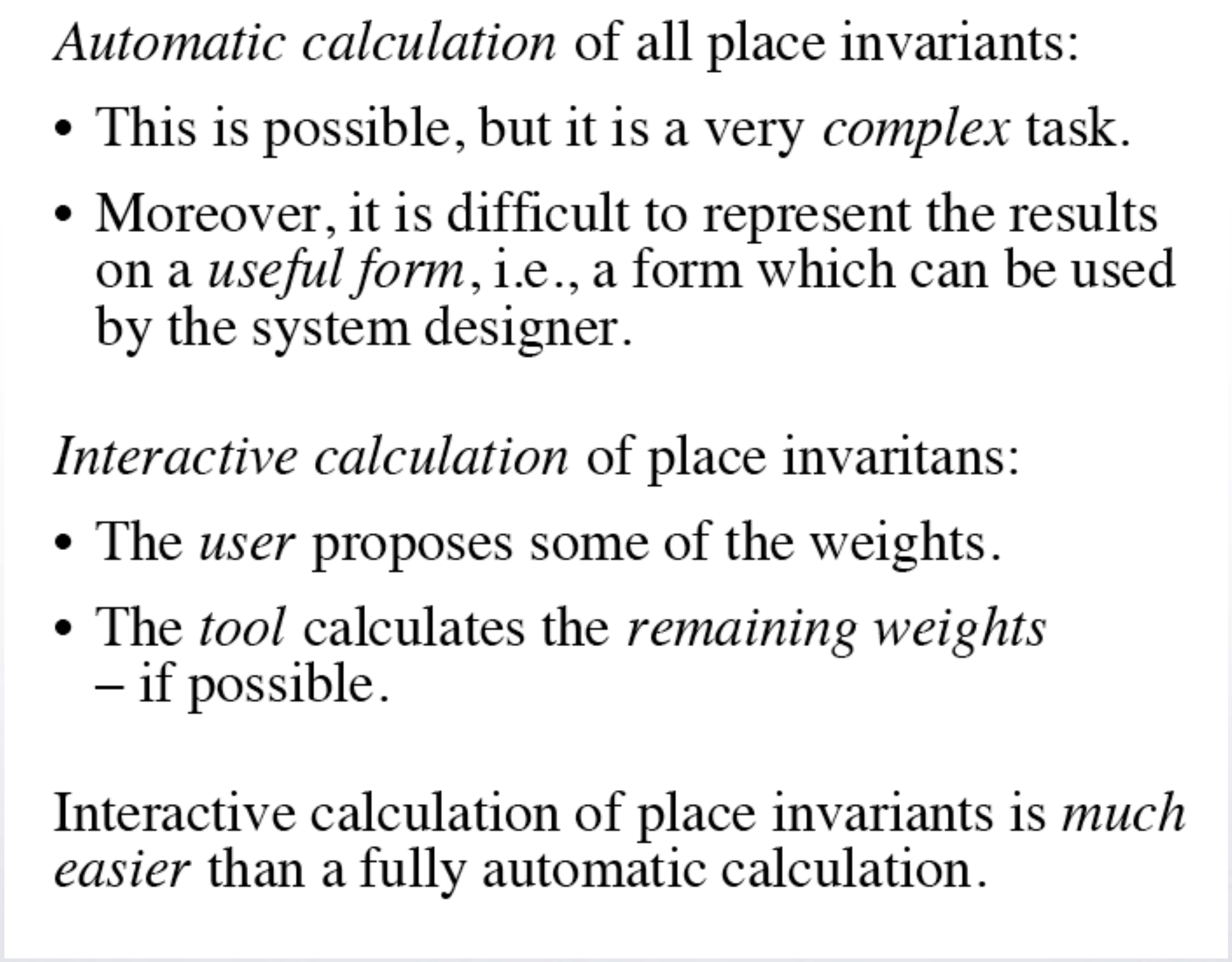
Then we check whether they are *fulfilled*.

- 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*.

Automating the invariant analysis



Automatic calculation of all place invariants:

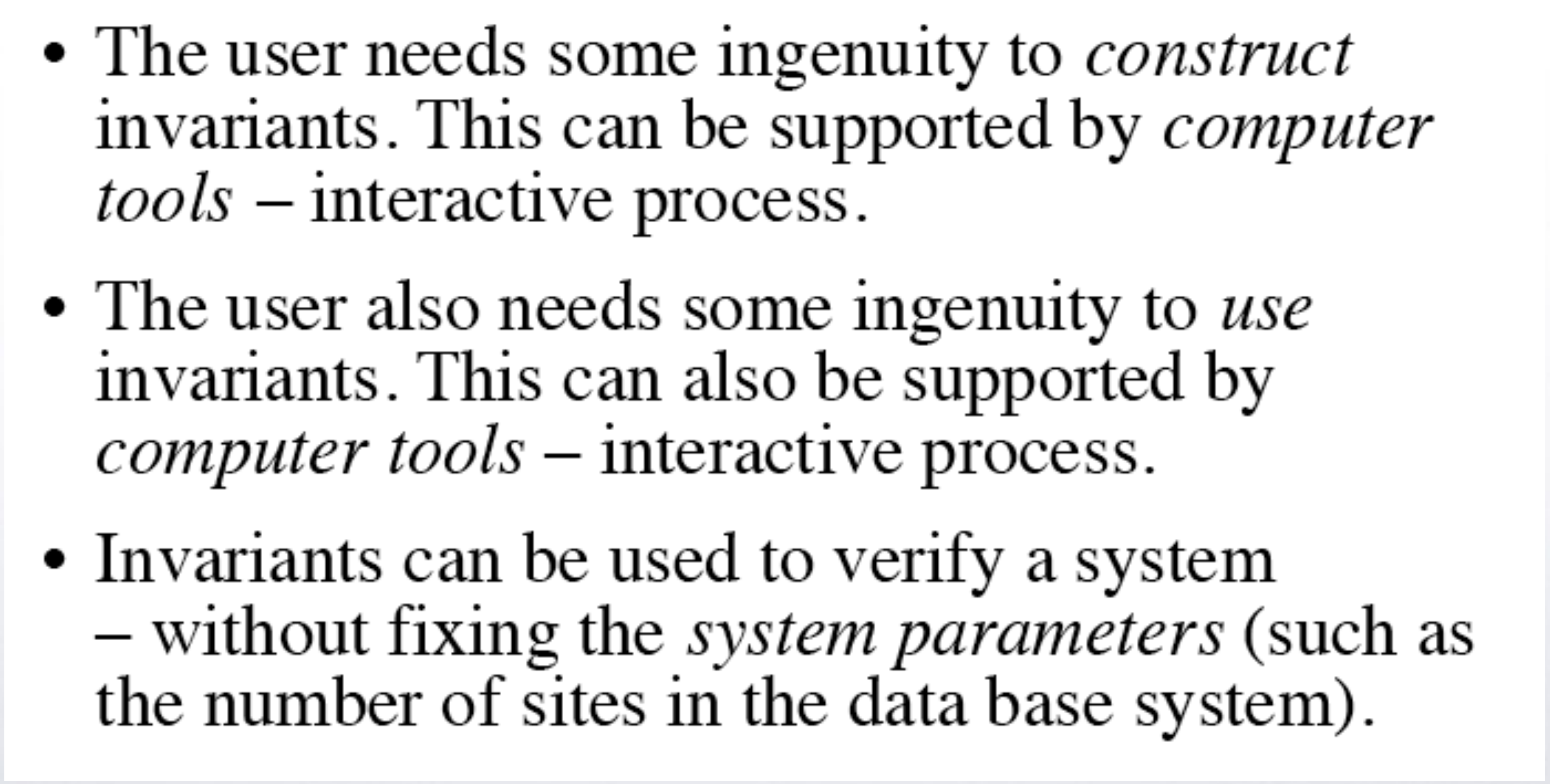
- This is possible, but it is a very *complex* task.
- Moreover, it is difficult to represent the results on a *useful form*, i.e., a form which can be used by the system designer.

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.

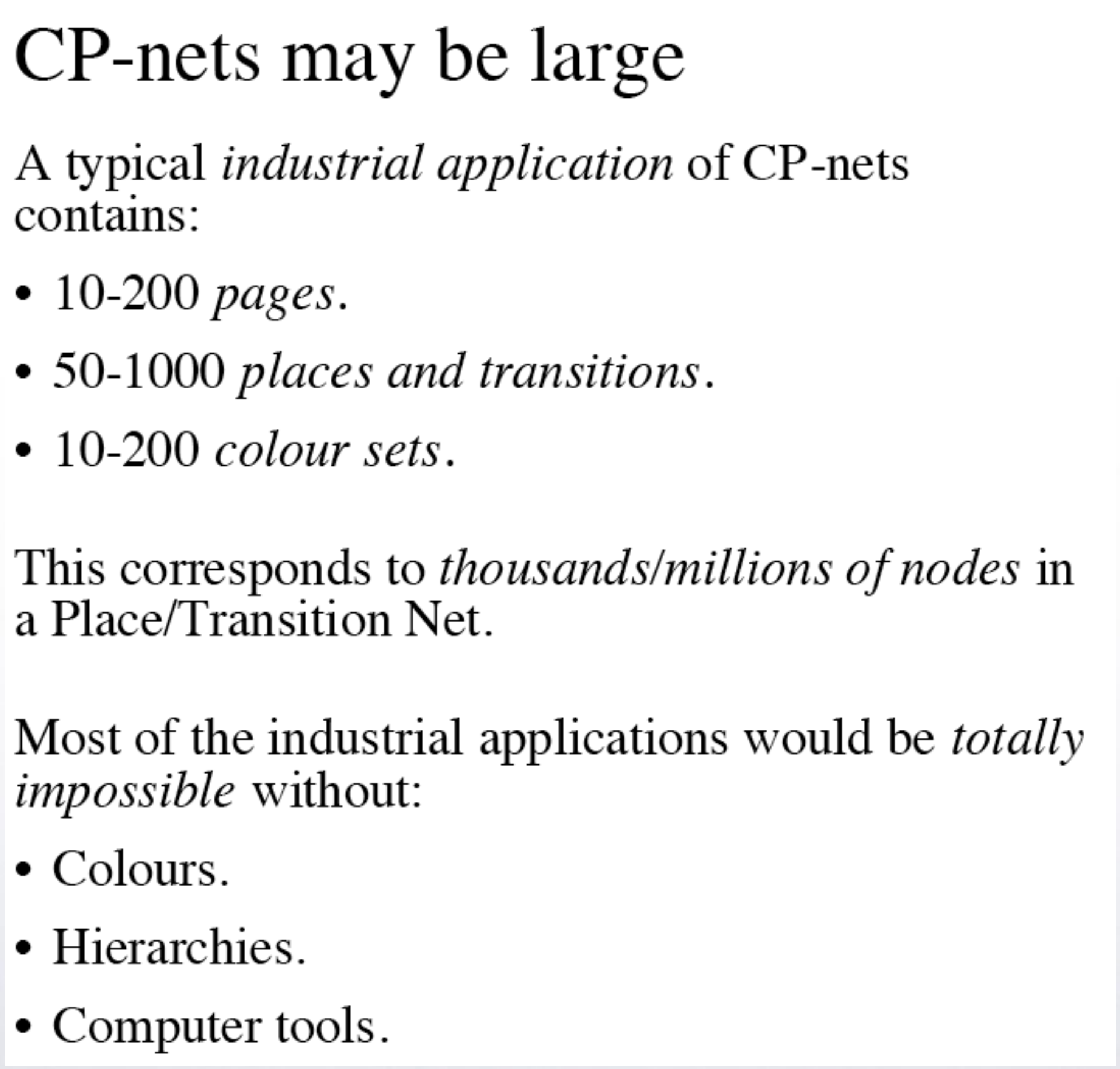
The invariant method in CPN



- 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).

Invariants are a very important feature in CPN Design. However, we should not expect to solve the design problem by just inserting invariant analysis.

Besides those inherent problems with invariants, the difficulty to apply this approach to large systems is still present.



CP-nets may be large

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

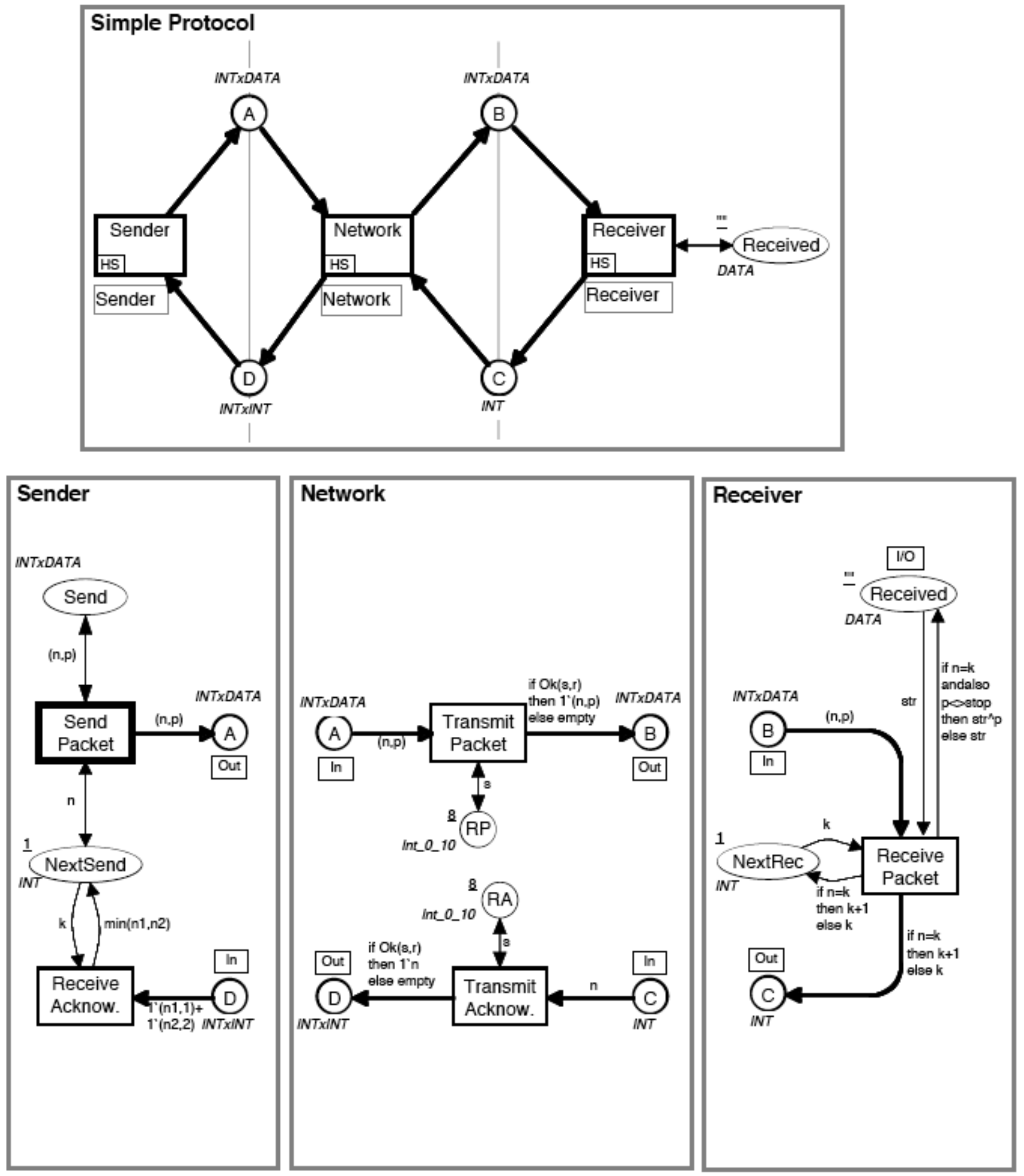
- 10-200 *pages*.
- 50-1000 *places and transitions*.
- 10-200 *colour sets*.

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.

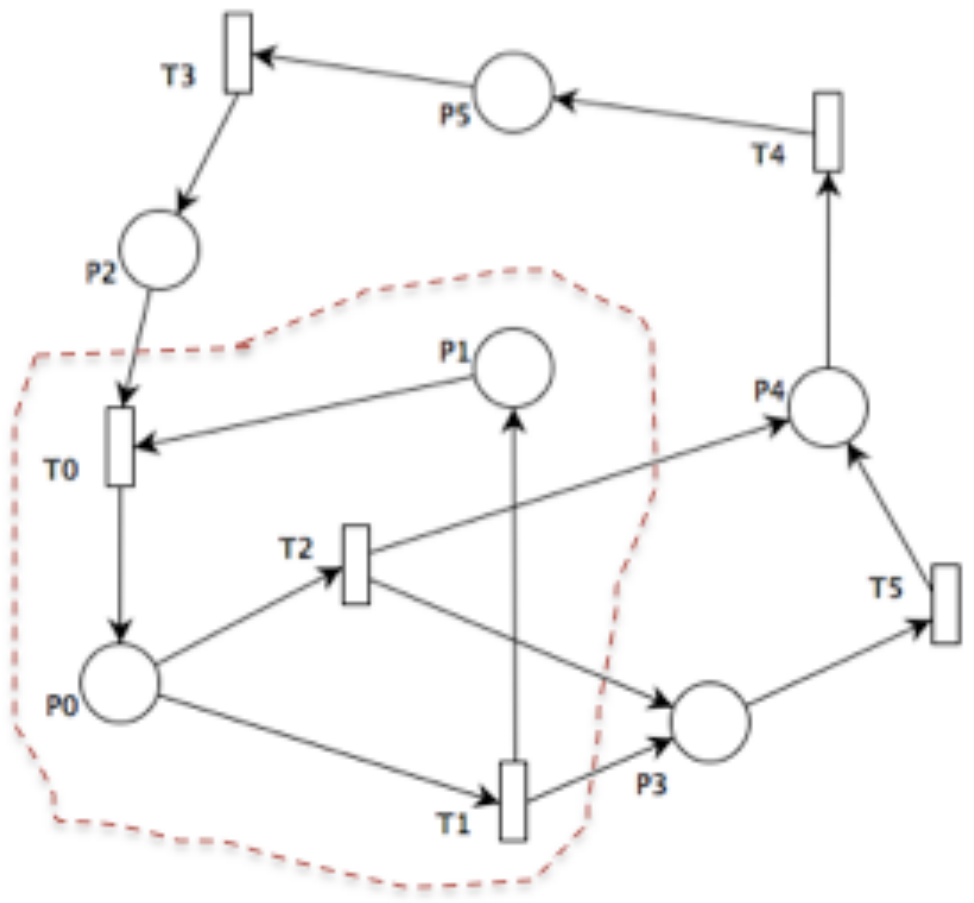
A hierarchical CP-net contains a number of *interrelated subnets*– called *pages*.



Hierarchy is not anything new and is actually connected with any kind of net, including the classical ones.

In design, hierarchy means to abstract the elements which properties are not relevant in an analysis phases.

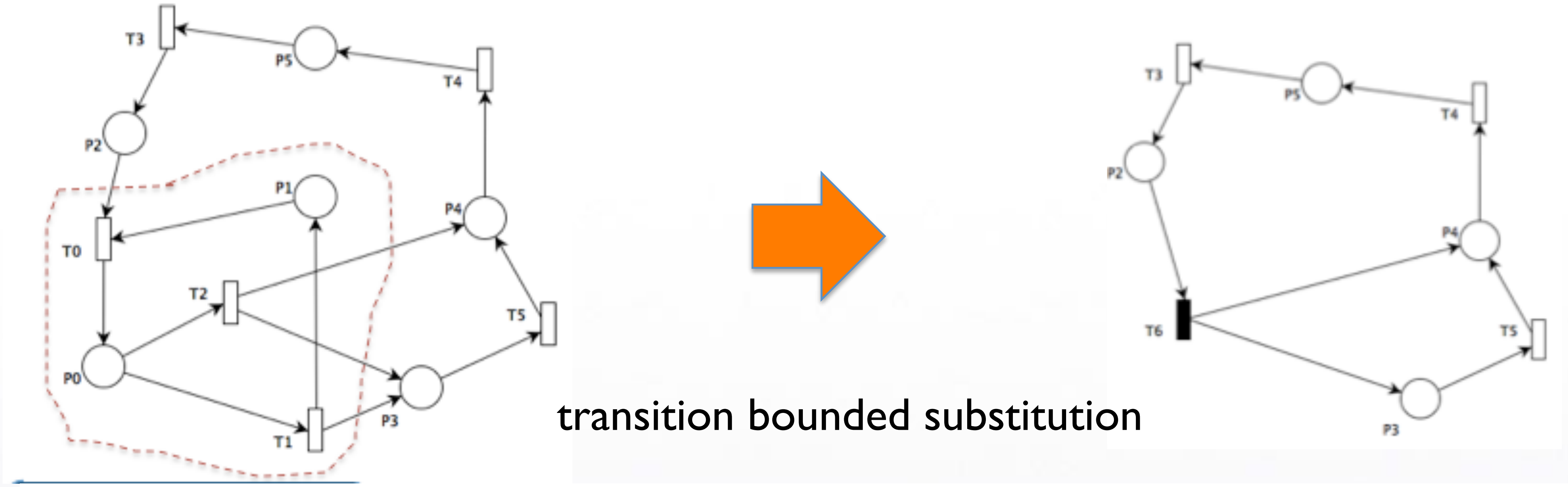
Hierarquia em redes clássicas



Definition 39

Seja uma estrutura de rede $N = (S, T; F)$. Seja $X = S \cup T$ e um sub-cojunto $Y \subseteq X$. Definimos uma borda de N como o conjunto $\partial(N) = \{y \in Y \mid \exists x \notin Y . x \in loc(y)\}$.

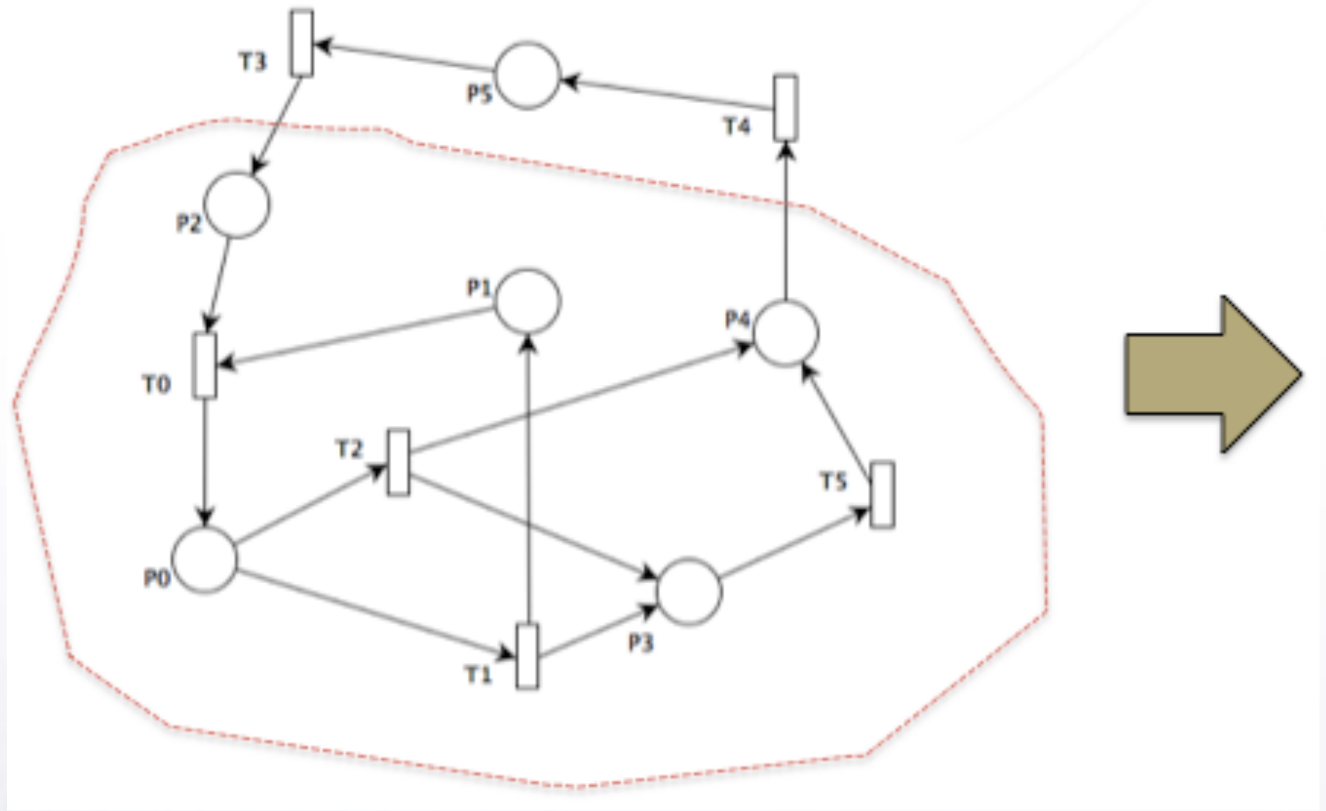
Substituição de uma sub-rede

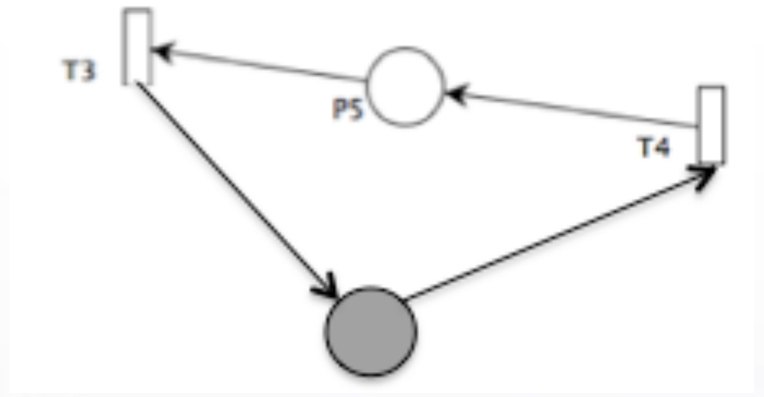


transition bounded substitution

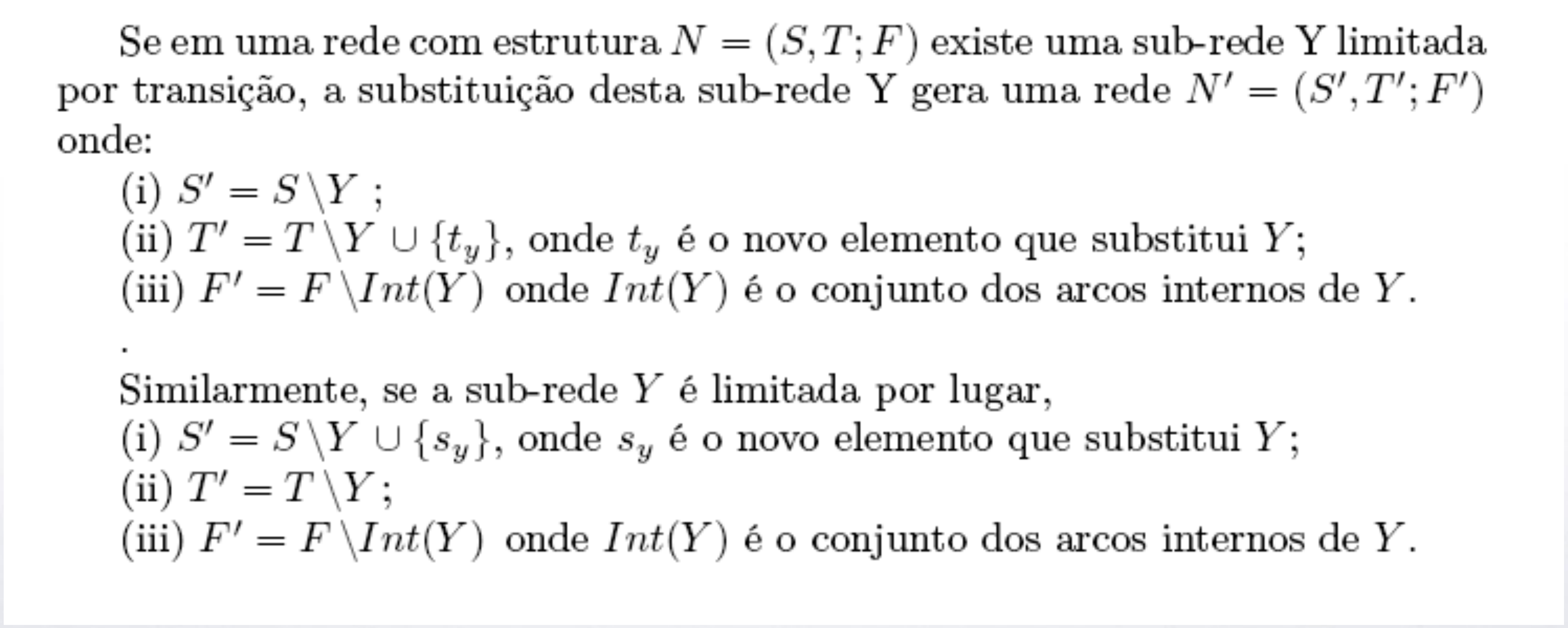
Definition 40

Um sub-conjunto de elementos Y da rede $N = (S, T; F)$ é dito limitado por lugar (place bounded) ou aberto, se e somente se $\partial(Y) \subseteq S$. Similarmente, um sub-conjunto Y desta rede é dito limitado por transição (transition bounded), se e somente se $\partial(Y) \subseteq T$.





place bounded substitution



Se em uma rede com estrutura $N = (S, T; F)$ existe uma sub-rede Y limitada por transição, a substituição desta sub-rede Y gera uma rede $N' = (S', T'; F')$ onde:

(i) $S' = S \backslash Y$;
(ii) $T' = T \backslash Y \cup \{t_y\}$, onde t_y é o novo elemento que substitui Y;
(iii) $F' = F \backslash Int(Y)$ onde $Int(Y)$ é o conjunto dos arcos internos de Y.

.

Similarmente, se a sub-rede Y é limitada por lugar,
(i) $S' = S \backslash Y \cup \{s_y\}$, onde s_y é o novo elemento que substitui Y;
(ii) $T' = T \backslash Y$;
(iii) $F' = F \backslash Int(Y)$ onde $Int(Y)$ é o conjunto dos arcos internos de Y.

Elementos próprios

Seja x_y um elemento genérico (instanciável por t_y ou por p_y). Este elemento é dito *próprio* se e somente se é limitado por transição (lugar), tem somente dois elementos de borda, com pelo menos um processo vivo entre eles.

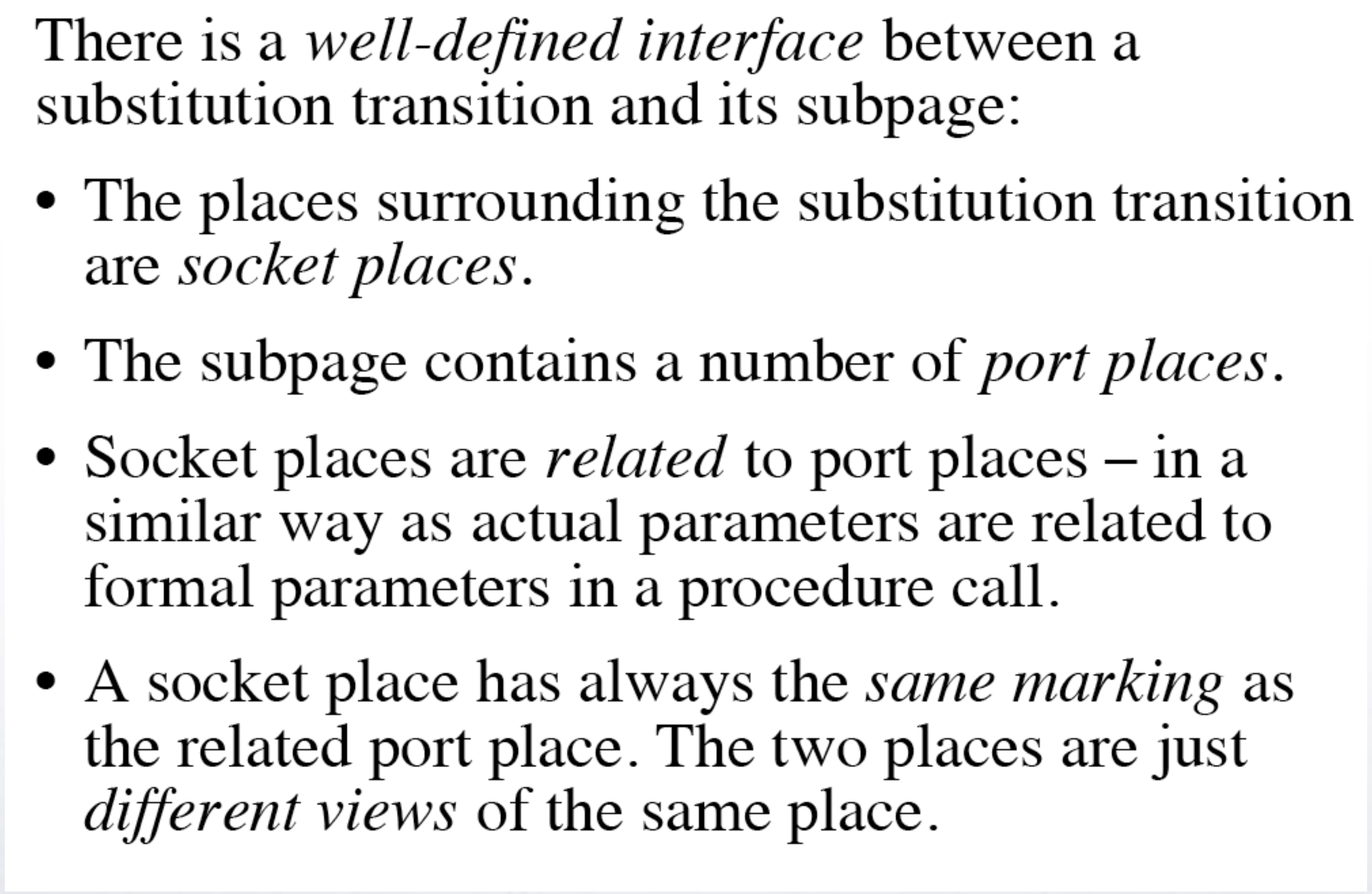
Se os elementos abstratos são próprios as propriedades da rede subjacente se conservam a menos de um termo aditivo. (J. R. Silva, On The Property Analysis of Abstract and Hierarchical Nets, to appear).

Hierarchy is a good abstraction feature. However, the real challenge is to associate that with the property analysis, so that the abstract net preserve the same properties than the expanded one.

The proper requirement is a key issue for that.

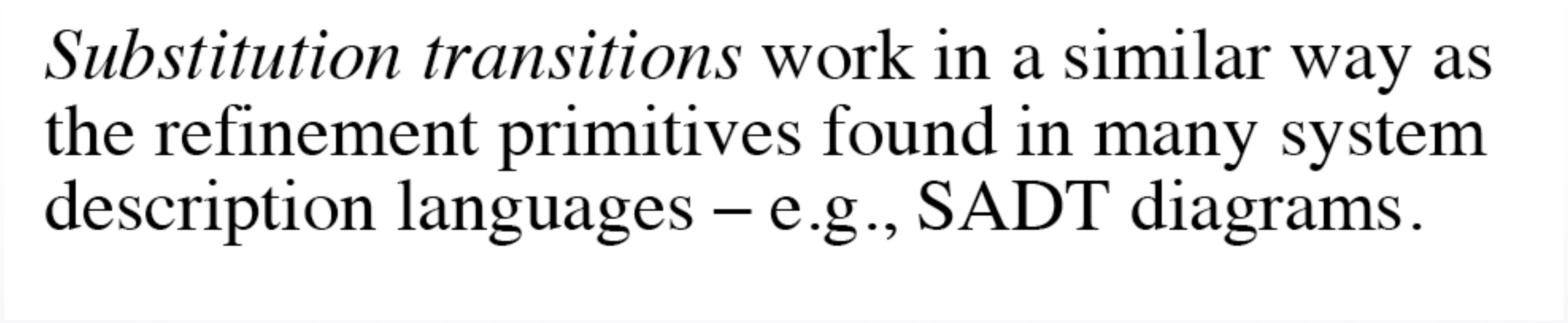
A page may contain one ore more *substitution transitions*.

- 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.

Fim