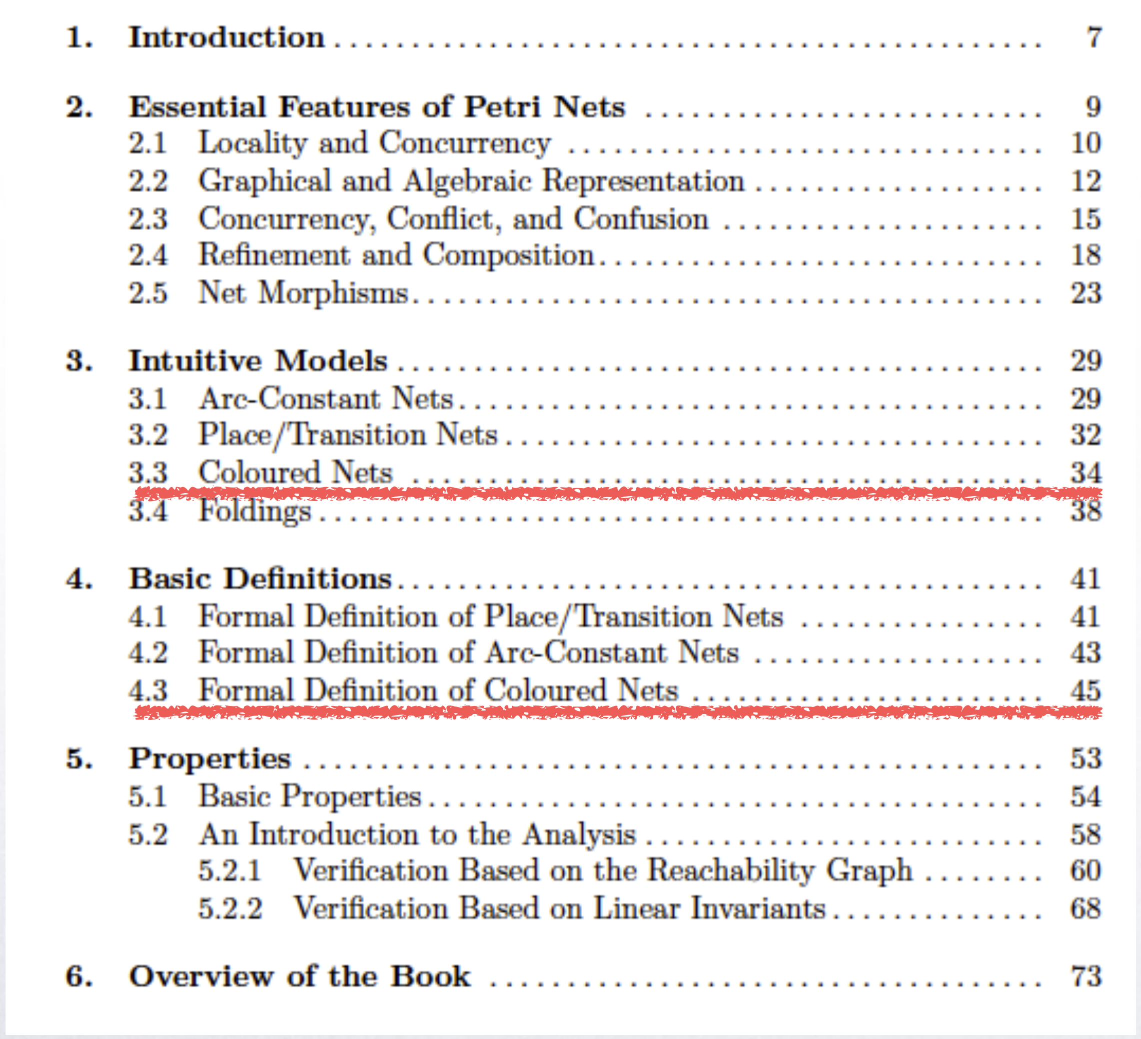
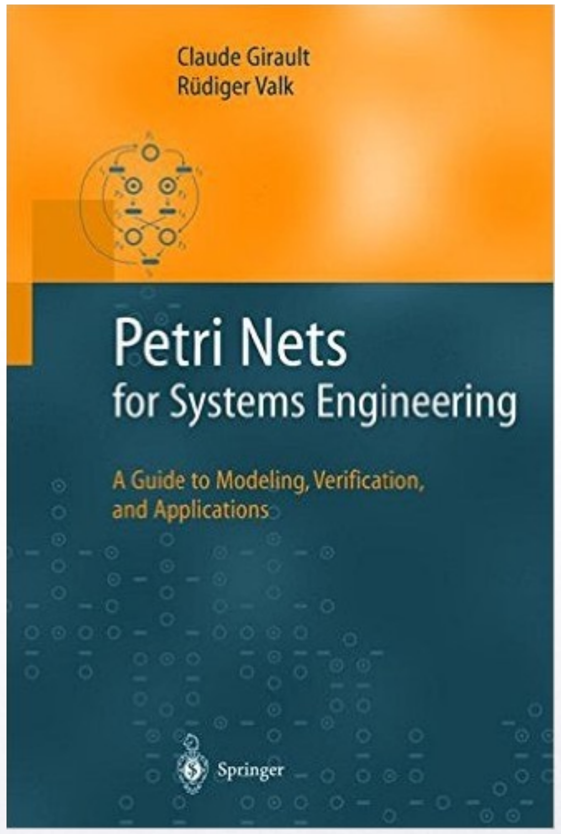
PMR 5237
Modelagem e Design de Sistemas Discretos em Redes de Petri

Aula 6: Redes Coloridas

Prof. José Reinaldo Silva
reinaldo@usp.br

Sincronismo com o livro texto




Claude Girault
Rüdiger Valk

Petri Nets
for Systems Engineering

A Guide to Modeling, Verification, and Applications

Springer

Na aula passada exploramos as simetrias no processo de modelagem de sistemas em redes de Petri, levando a um processo de fatoração ou dobramento das redes, resultando em redes de alto nível HLPN ou em redes coloridas.

Last week...

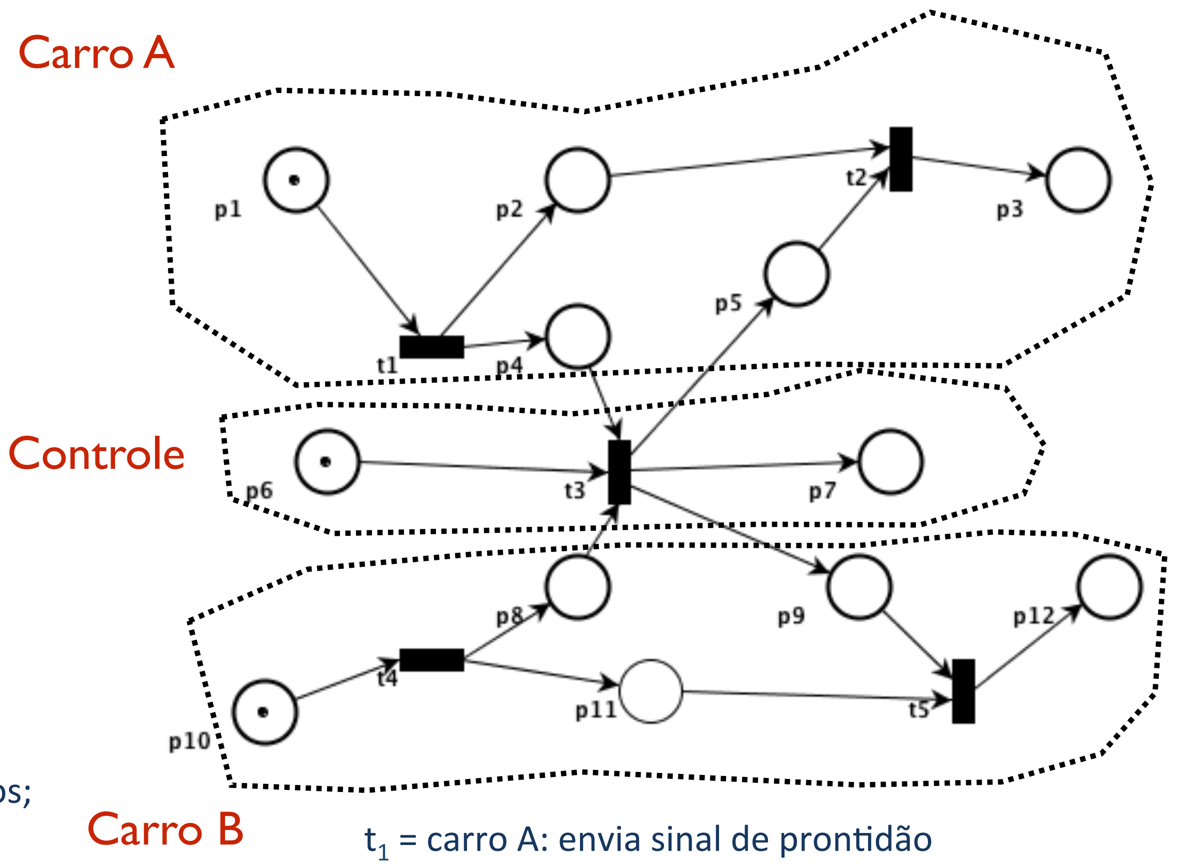
Folding X Unfolding

A fatoração (folding, ou dobramento) pode ser incluída nos métodos de modelagem, assim como sua operação inversa, o unfolding.

$$\begin{tikzcd}
\text{Rede P/T} \arrow[r, shift left, "\text{Fatoração}"] & \text{Rede Quociente} \arrow[l, shift left, "\text{Unfolding}"]
\end{tikzcd}$$

Explorar simetrias é um atributo muito poderoso na modelagem de sistemas. Podemos, por exemplo rever alguns dos exercícios discutidos, buscando as simetrias do problema...

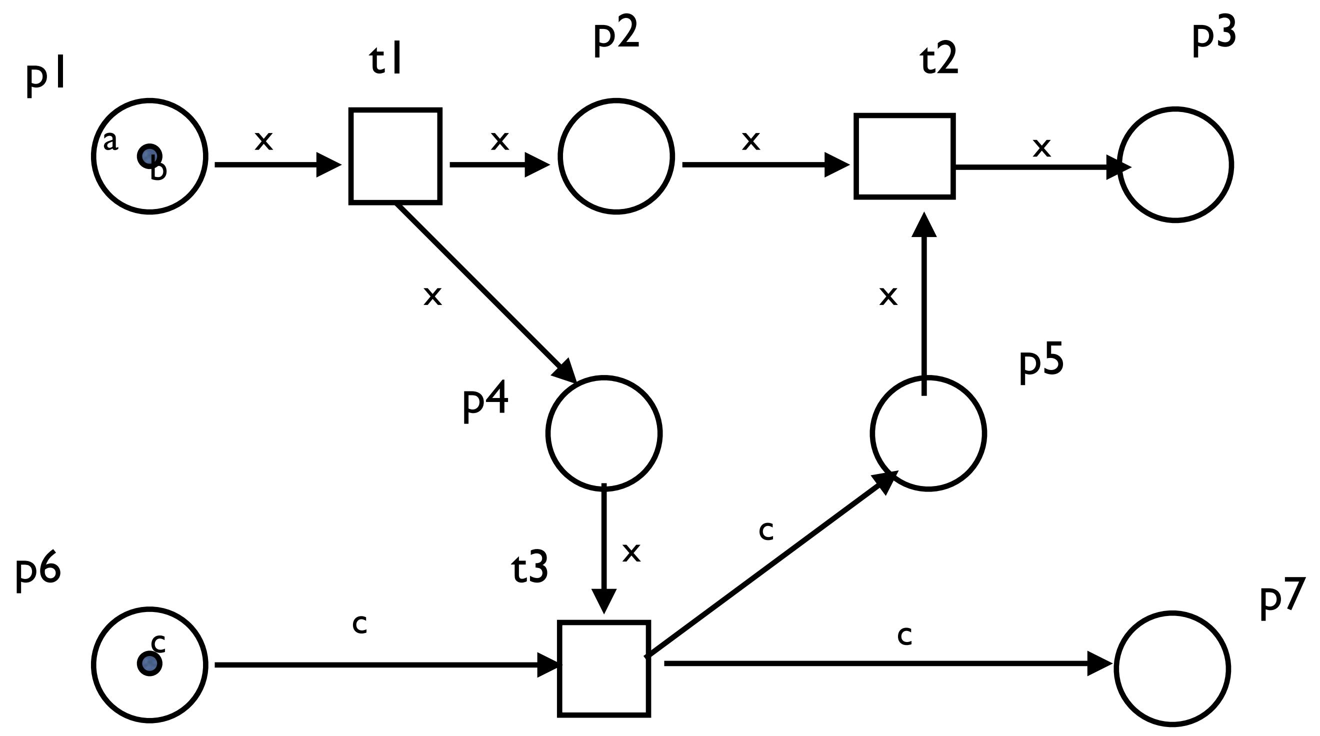
Exemplo da largada de fórmula I

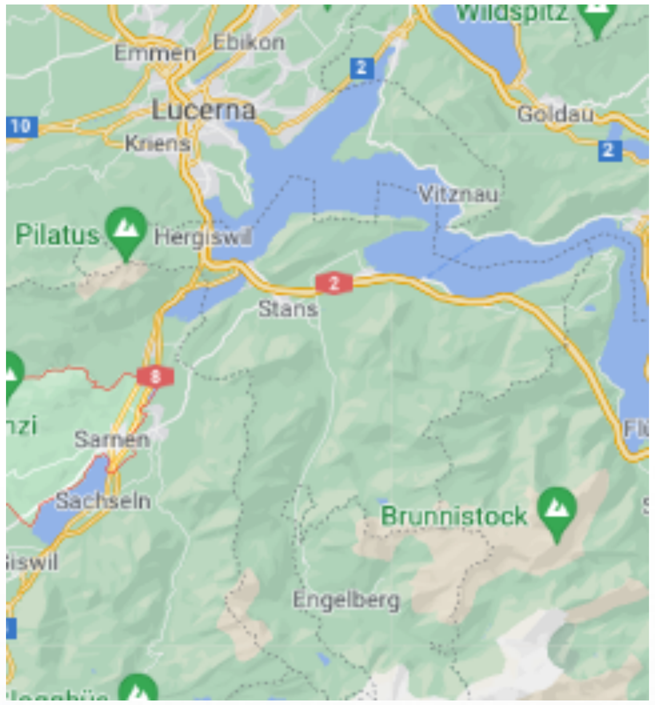


p_1= carro A: preparando-se para começar;
p_2= carro A: esperando o sinal de largada;
p_3= carro A: correndo;
p_4= sinal de prontidão do carro A enviado;
p_5= sinal de largada para o carro A enviado;
p_6= operador: esperando sinal de prontidão dos pilotos;
p_7= operador: sinal de largada enviado;
p_8= sinal de prontidão do carro B enviado;
p_9= sinal de largada para o carro B enviado;
p_{10}= carro B: preparando-se para começar;
p_{11}= carro B: esperando o sinal de largada;
p_{12}= carro B: correndo;

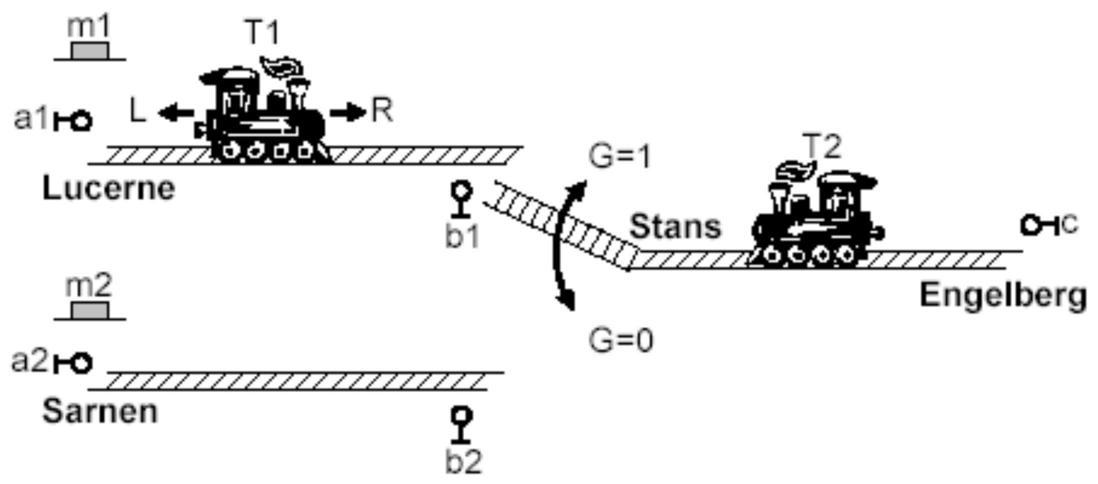
t_1 = carro A: envia sinal de prontidão
t_2 = carro A: acelera
t_3 = operador: manda sinal de largada
t_4 = carro B: envia sinal de prontidão
t_5 = carro B: acelera

Cars = {a,b}
Cont = c



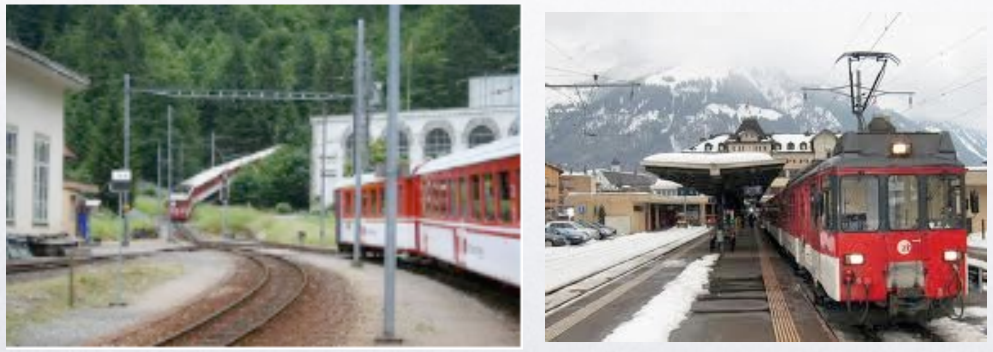


Trecho do mapa da Suíça



Sarnen

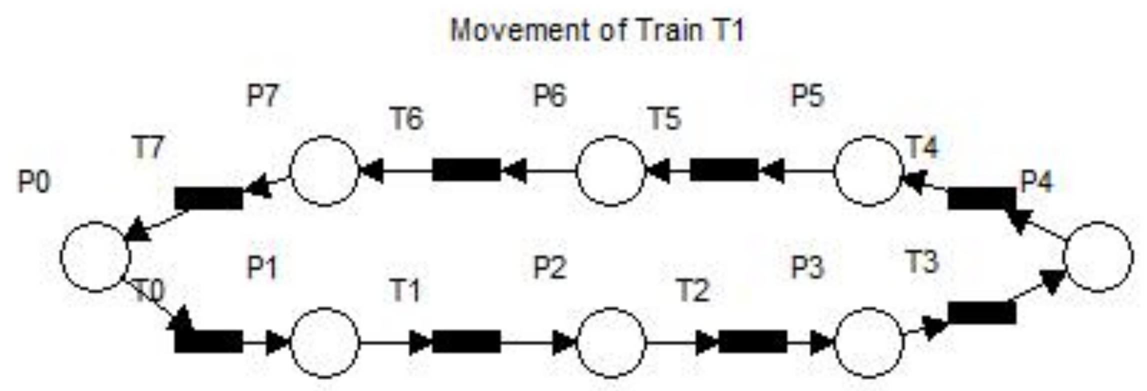
Monte Titlis, Engelsberg



Stans

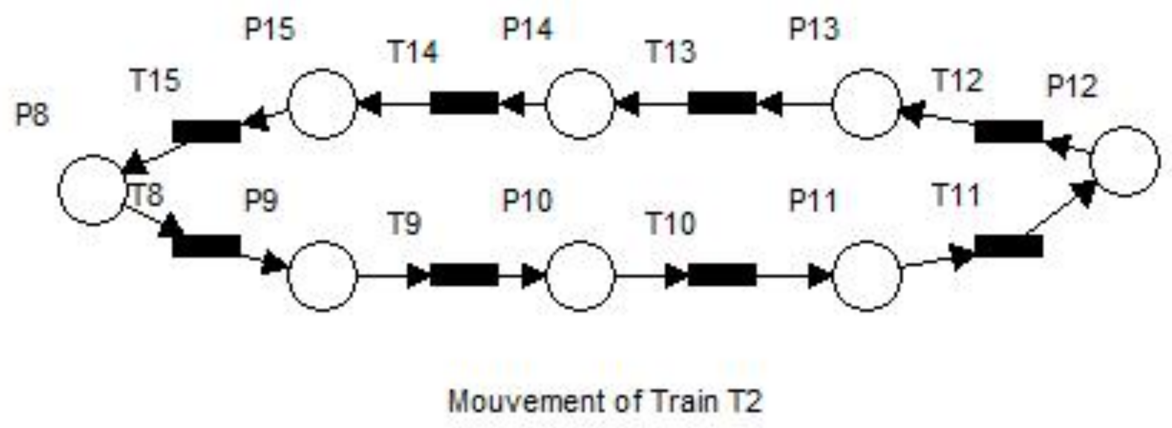
Movimento do trem T1

P1 – trem PT1 no ponto a1 (Lucerne);
P2 – trem T1 indo de Lucerne para Stans;
P3 – trem T1 chega em stans (detectado pelo sensor b1)
P4 – trem T1 no trecho unificado Stans Engelberg
P5 – trem T1 chega em Engelberg;
P6 – trem T1 indo de Engelberg para Stans (trecho unificado);
P7 – trem T1 chega no gate 1 (não há detecção por b1);
P8 – trem T1 indo de Stans para Lucerne;



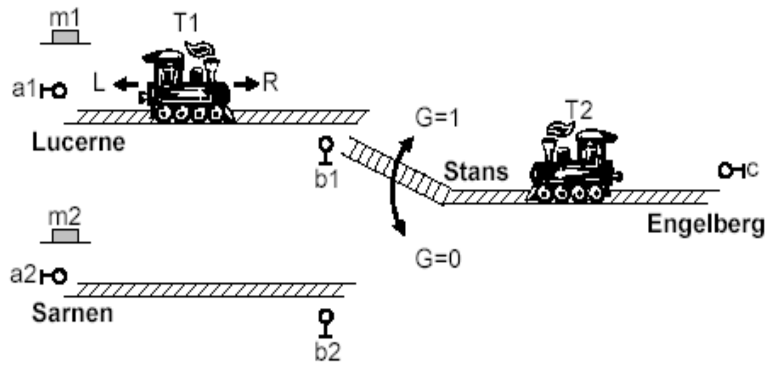
Movimento do trem T2

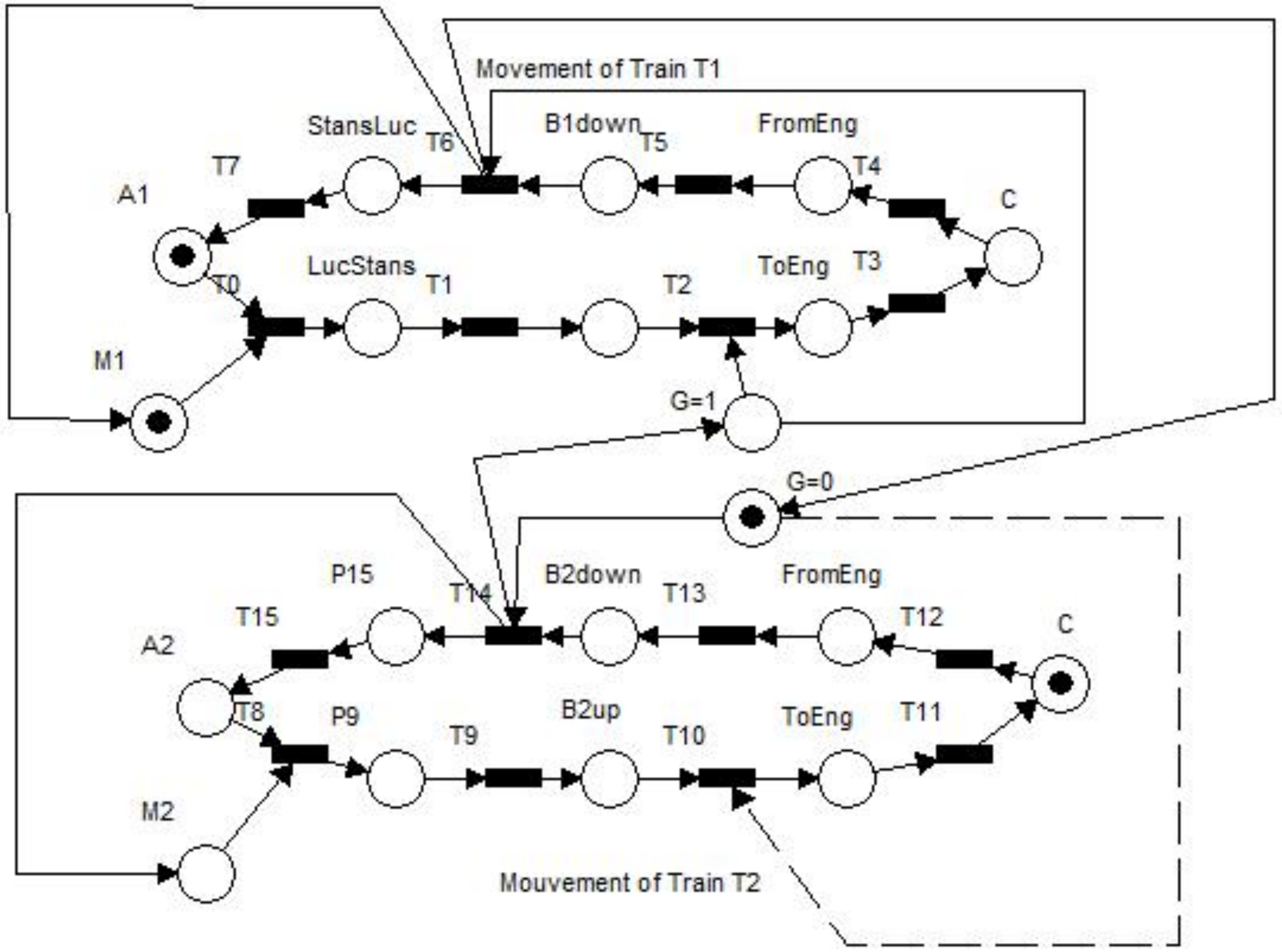
P9 – trem T2 no ponto C (Engelberg);
P10 – trem T2 indo de Engelberg para Stans;
P11 – trem T2 chega em Stans (não detectado pelo sensor b2)
P12 – trem T2 indo de Stans para Sarnen;
P13 – trem T2 chega em Sarnen;
P14 – trem T2 indo de Sarnen para Stans ;
P15 – trem T2 chega no gate 1 (Stans) (detectado pelo sensor b2);
P16 – trem T2 indo de Stans para Engelberg;

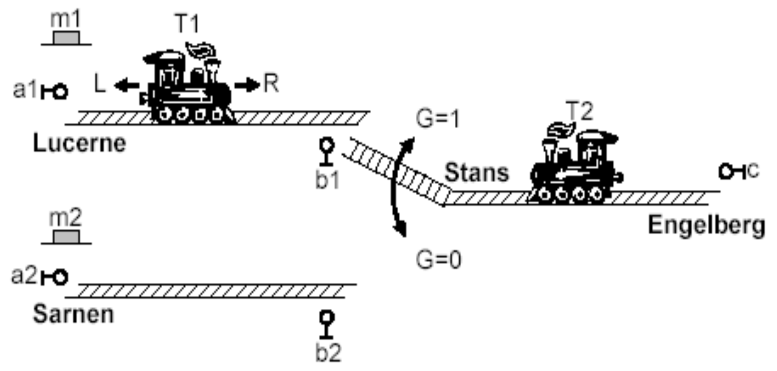


O uso de extensões

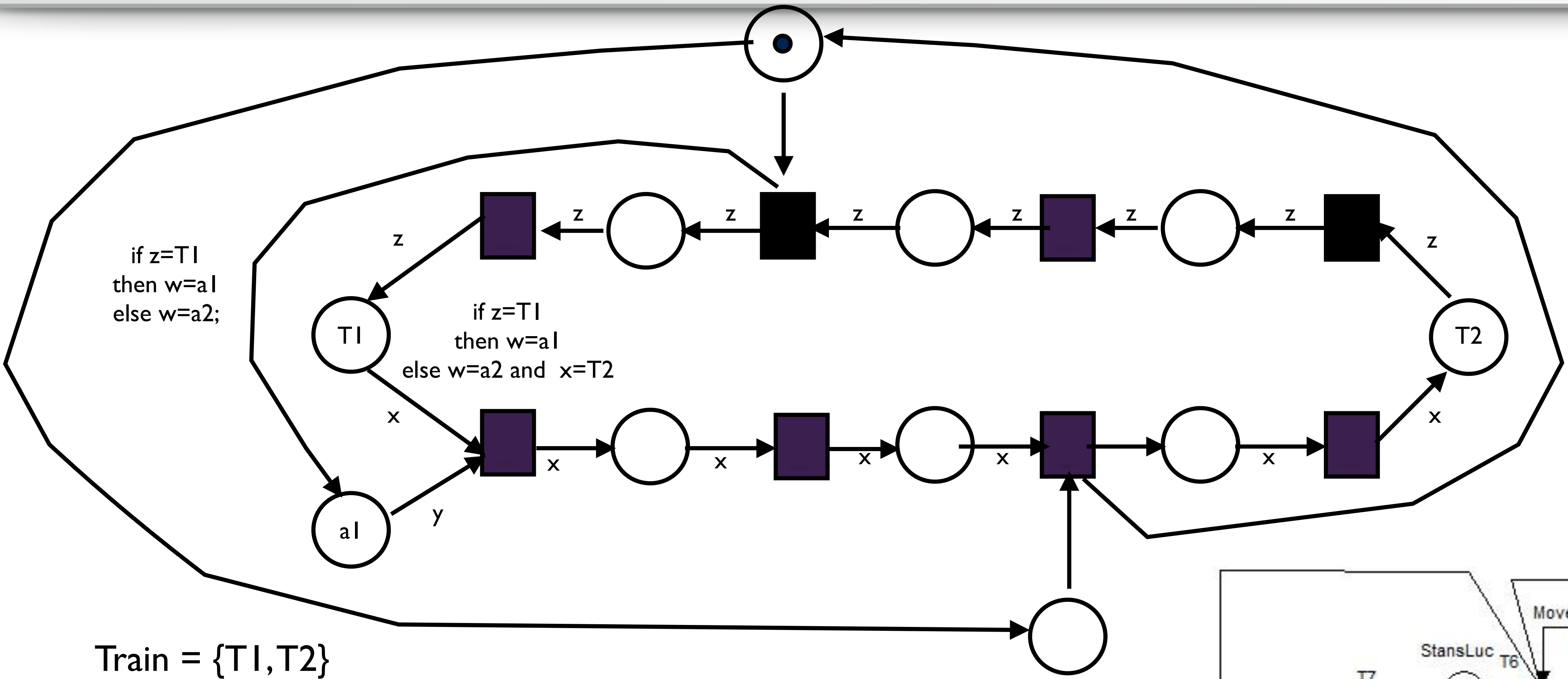
Ao lado mostramos o mesmo modelo, agora utilizando arcos especiais que propagam apenas a informação sobre a marcação mas não fazem com que a marca se propague pela rede. Estes arcos se chamam "gates" (não confundir com o desvio da via férrea colocado anteriormente).



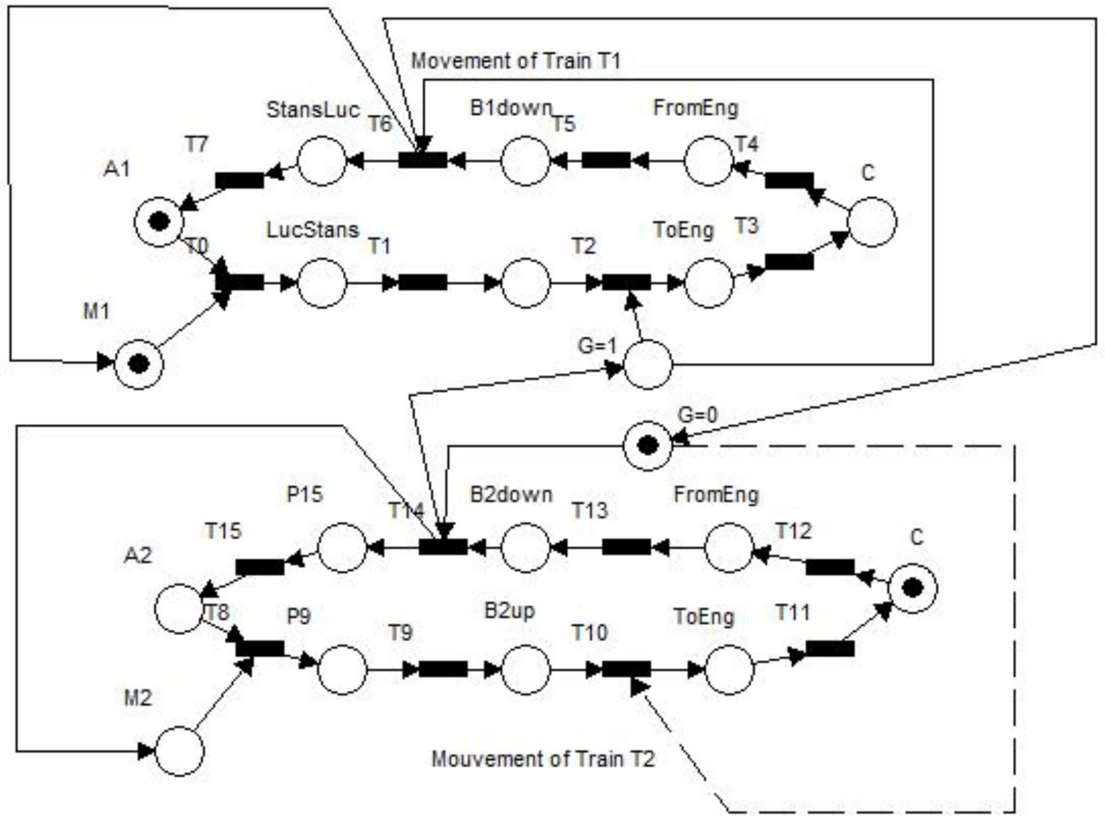




Embora seja "menos visual", a simetria nesse caso está ligada à similaridade no movimento dos trens, com chaveamento (automatizado) em Stans.



Train = {T1,T2}
op_sign = {a1, a2}
gate = {0, 1}
x, z in Train
w, y in op_sign



Sempre vale a pena o dobramento?

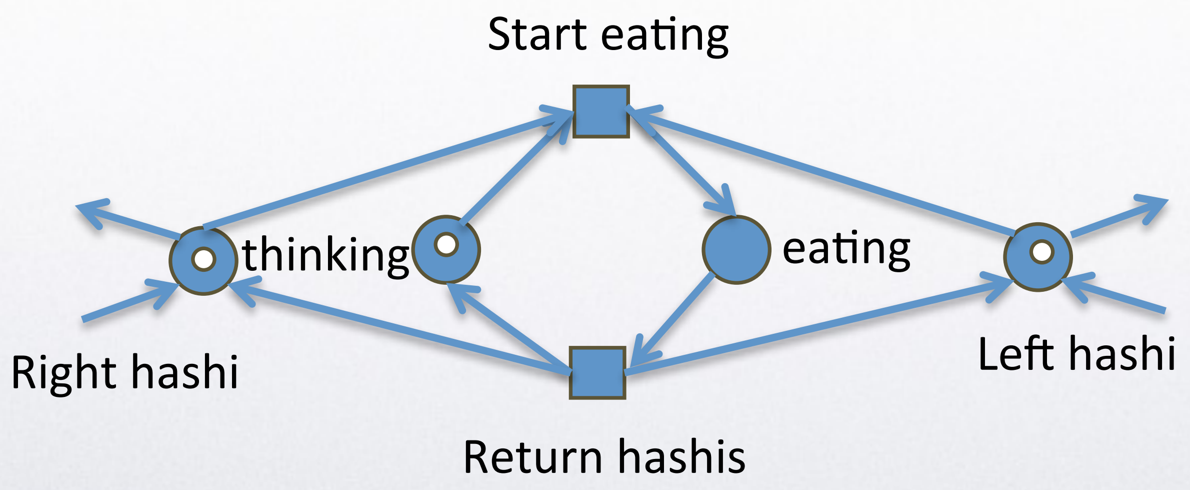
$$\begin{tikzcd}\text{Rede P/T} \arrow[r, shift left, "\text{Fatoração}"] & \text{Rede de Alto Nível ou Colorida} \arrow[l, shift left, "\text{Unfolding}"]\end{tikzcd}$$

Você conseguiria sintetizar uma rede de alto nível (ou colorida) diretamente do enunciado do problema?

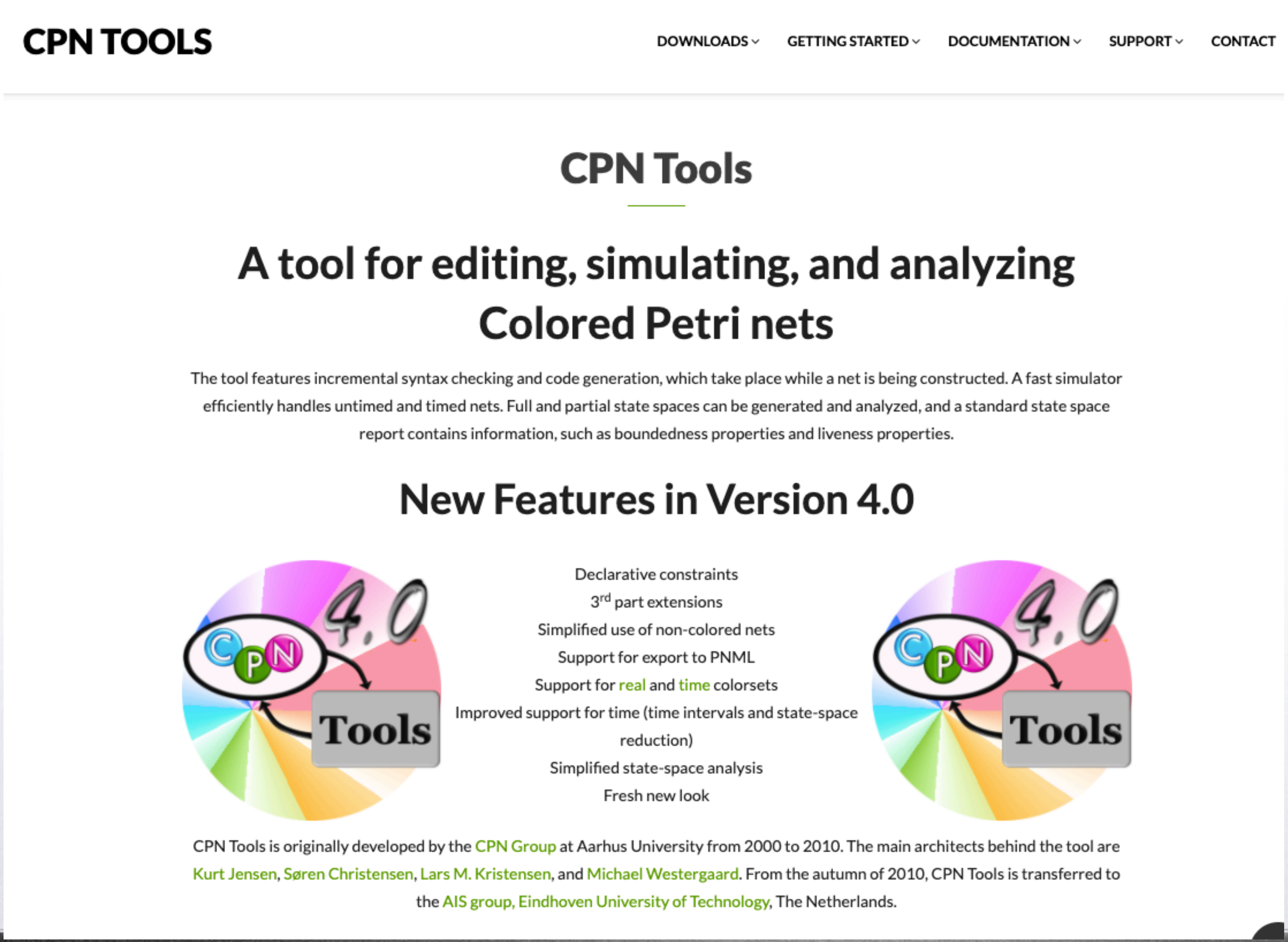
Tente fazer isso com um dos problemas mostrados anteriormente, mesmo já tendo visto a solução.

Modelagem e simetria

Certamente, uma forma de olhar o problema é procurar, logo de início, o relacionamento entre TODOS os seus elementos constituintes, como feito no slide anterior. Mas é possível também olhar cada um dos elementos composicionais, especialmente aqueles que apresentam propriedades reptetitivas. No exemplo dos filósofos, se olharmos cada um dos filósofos, temos:



Redes Coloridas



CPN TOOLS

DOWNLOADS GETTING STARTED DOCUMENTATION SUPPORT CONTACT

CPN Tools

A tool for editing, simulating, and analyzing Colored Petri nets

The tool features incremental syntax checking and code generation, which take place while a net is being constructed. A fast simulator efficiently handles untimed and timed nets. Full and partial state spaces can be generated and analyzed, and a standard state space report contains information, such as boundedness properties and liveness properties.

New Features in Version 4.0

Declarative constraints
3rd part extensions
Simplified use of non-colored nets
Support for export to PNML
Support for real and time colorsets
Improved support for time (time intervals and state-space reduction)
Simplified state-space analysis
Fresh new look

CPN Tools is originally developed by the CPN Group at Aarhus University from 2000 to 2010. The main architects behind the tool are Kurt Jensen, Søren Christensen, Lars M. Kristensen, and Michael Westergaard. From the autumn of 2010, CPN Tools is transferred to the AIS group, Eindhoven University of Technology, The Netherlands.

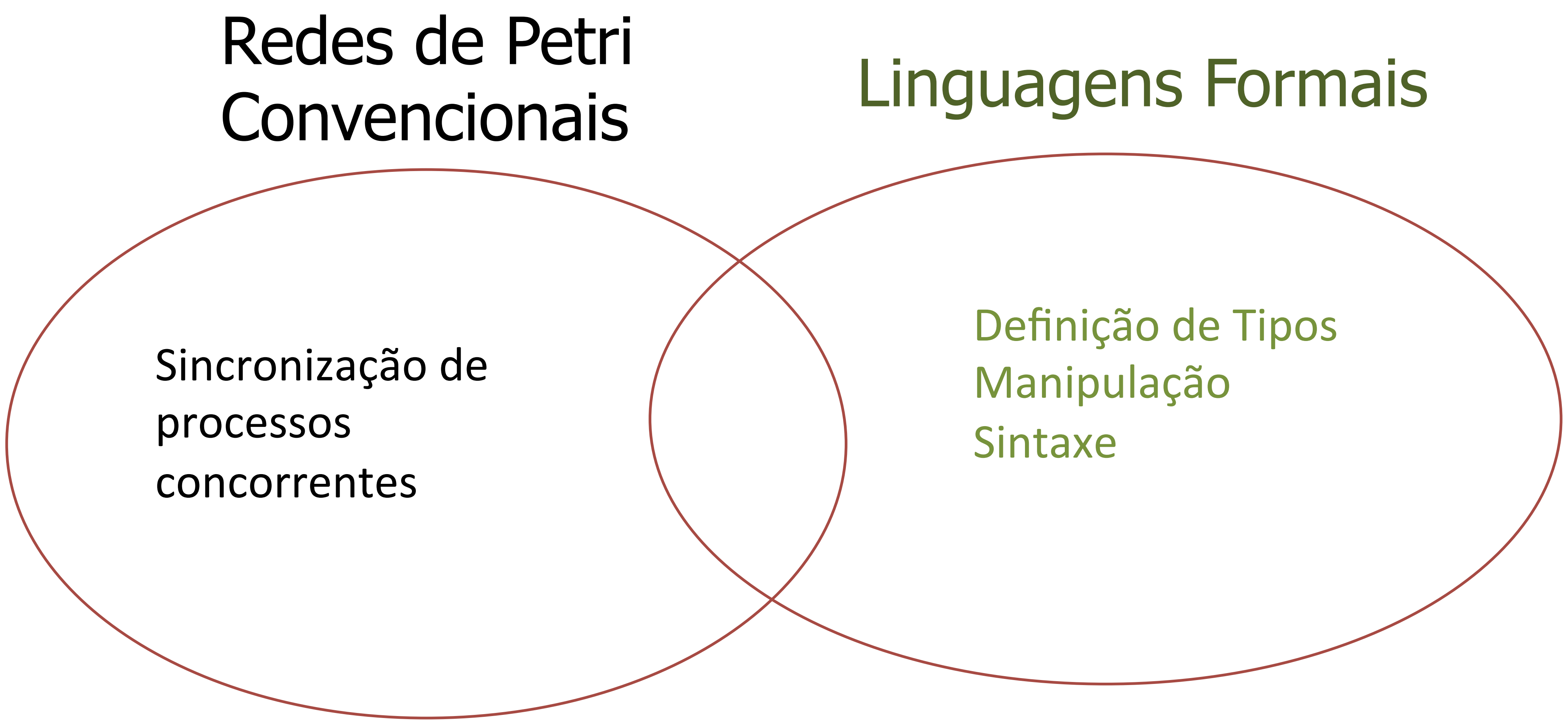
cpntools.org

Histórico das redes CPN

As redes coloridas surgiram nos anos 80 conjugando a representação gráfica das redes de Petri com o Standard ML, que representa tipos e cores.

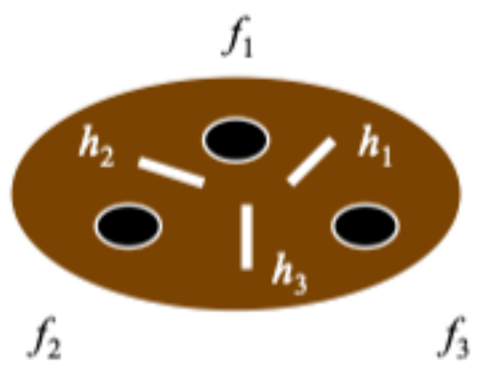
No final dos anos 80 e princípio dos anos 90 surgiu o ambiente Dsign CPN proposto pelo mesmo grupo de Ahus, Dinamarca (Kurt Jensen).

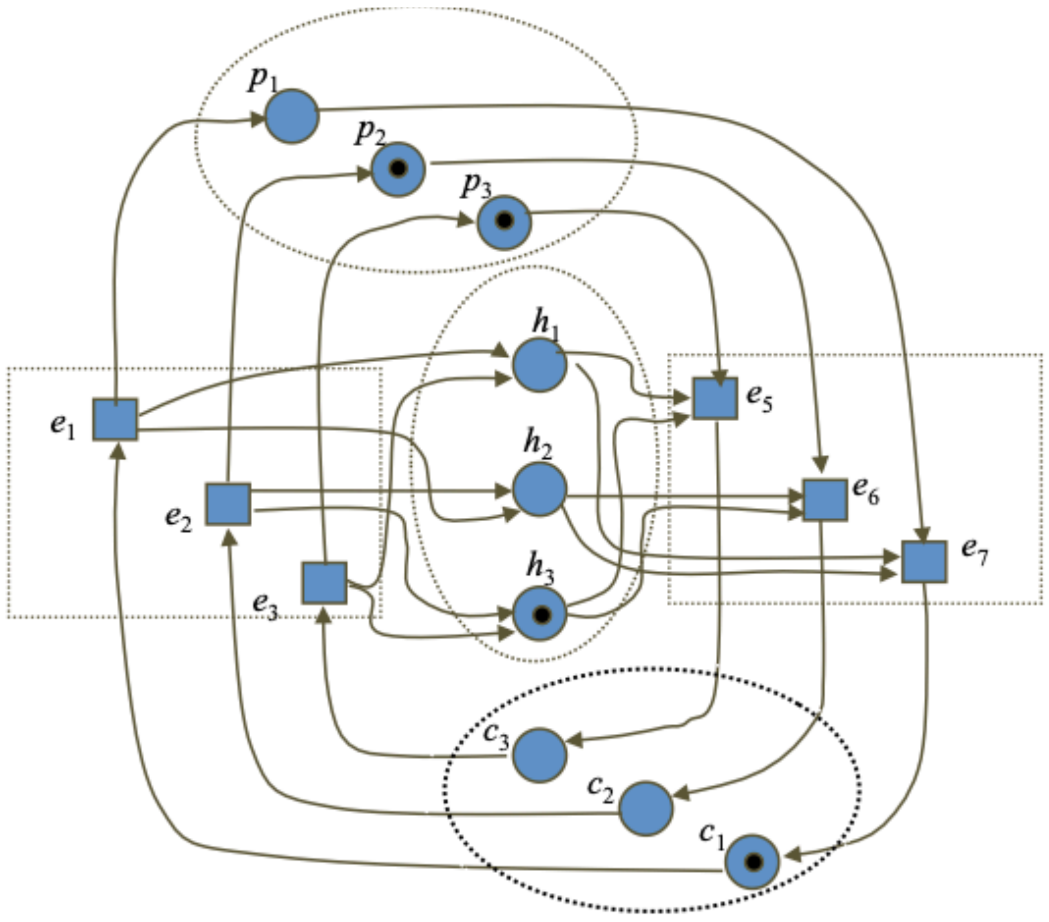
A idéia é simplesmente ter um formalismo mais abstrato

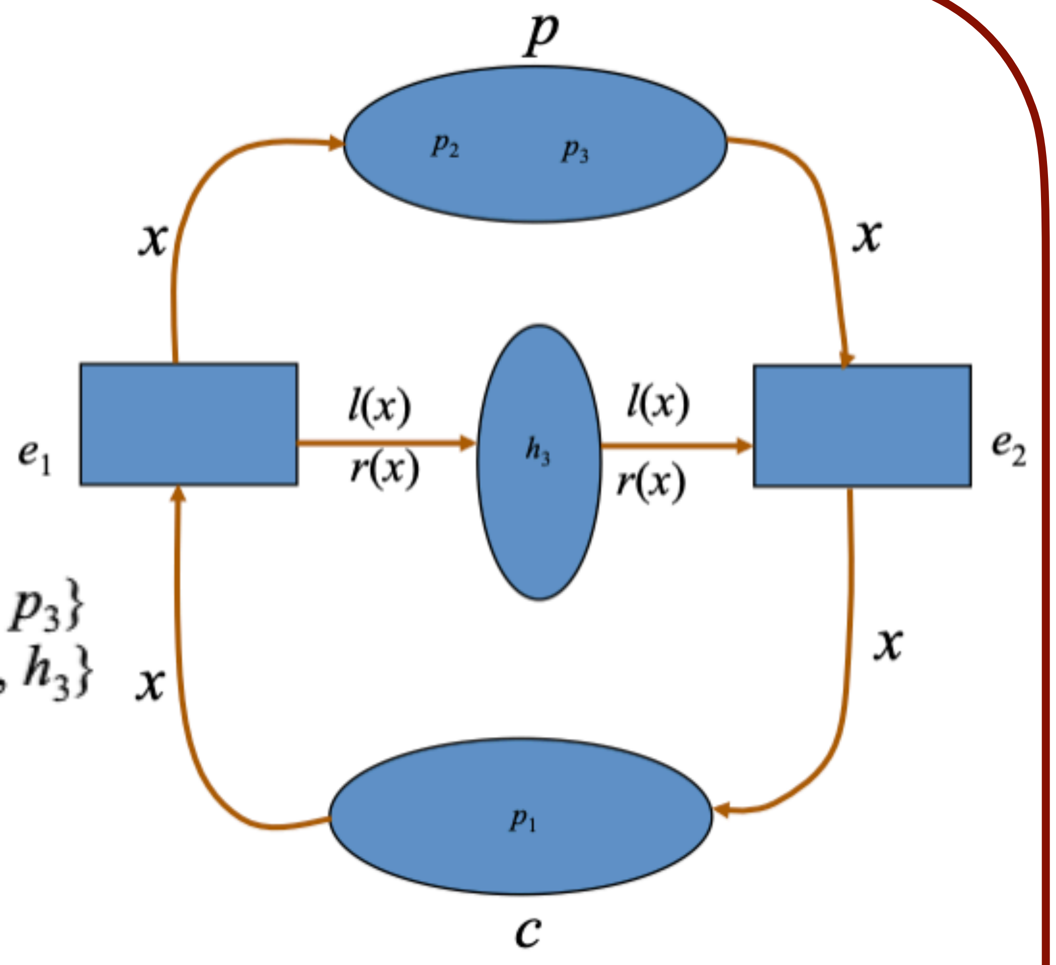

Redes de Petri Convencionais
Linguagens Formais
Sincronização de processos concorrentes
Definição de Tipos
Manipulação
Sintaxe

Kurt Jensen, An Introduction to the Practical Use of Coloured Petri Nets, Lect. Notes in Comp. Science 1492, 1998.

O exemplo dos filósofos

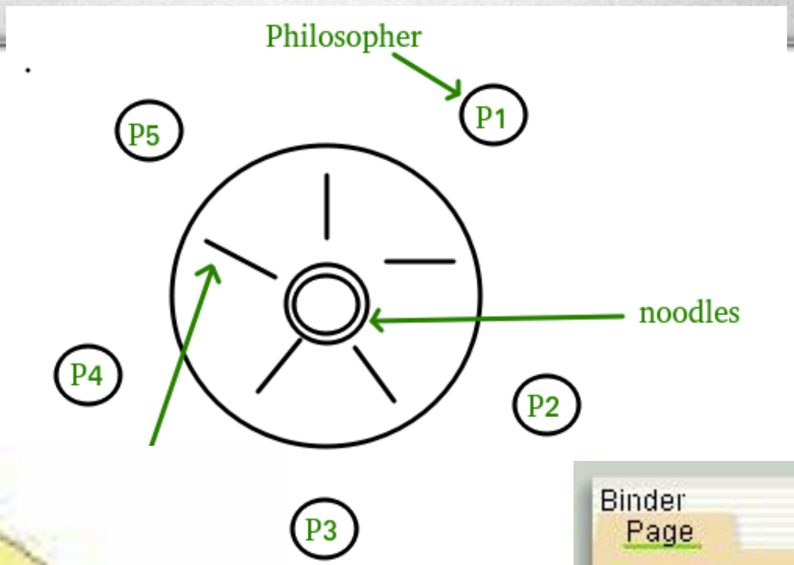


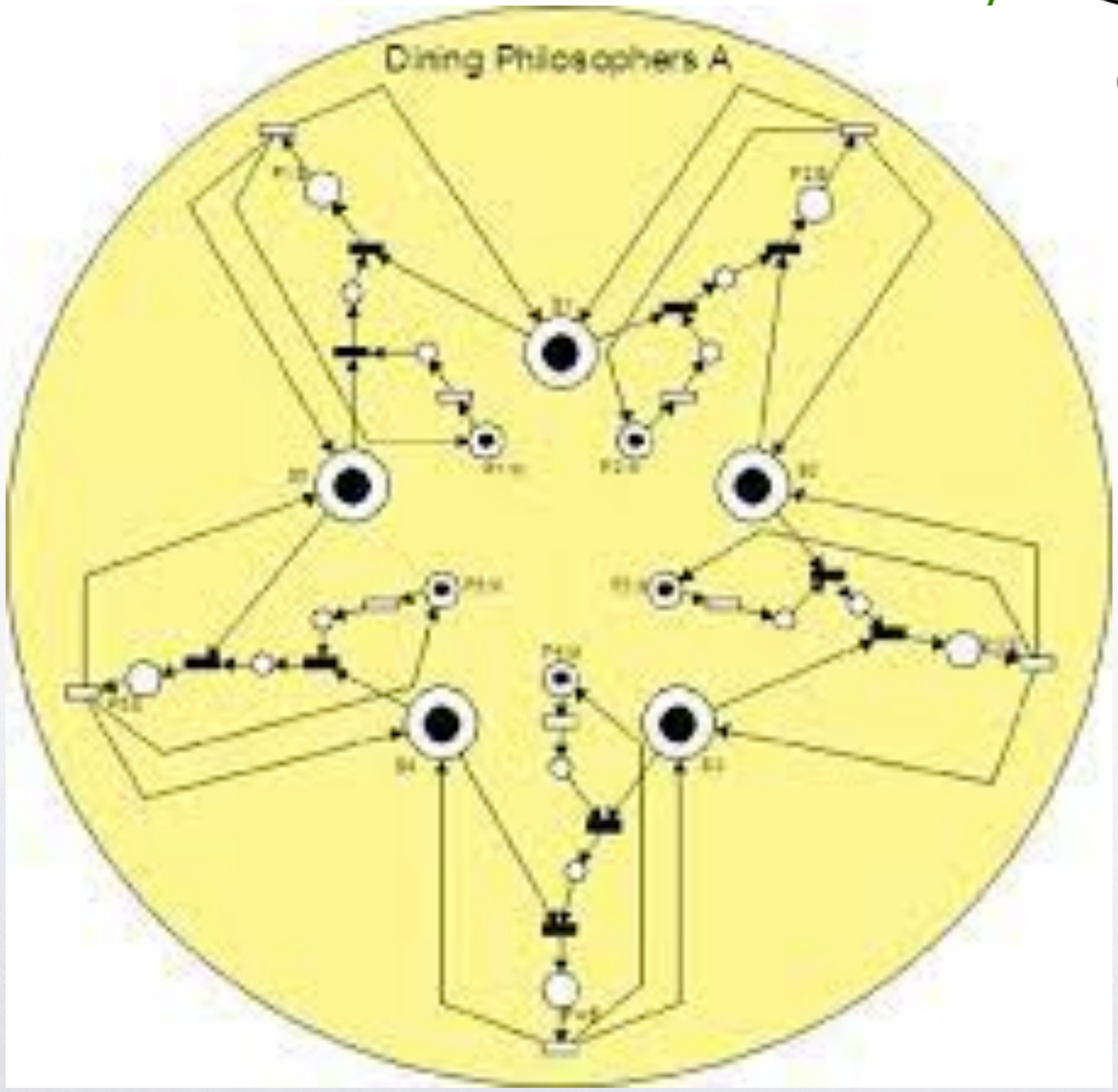


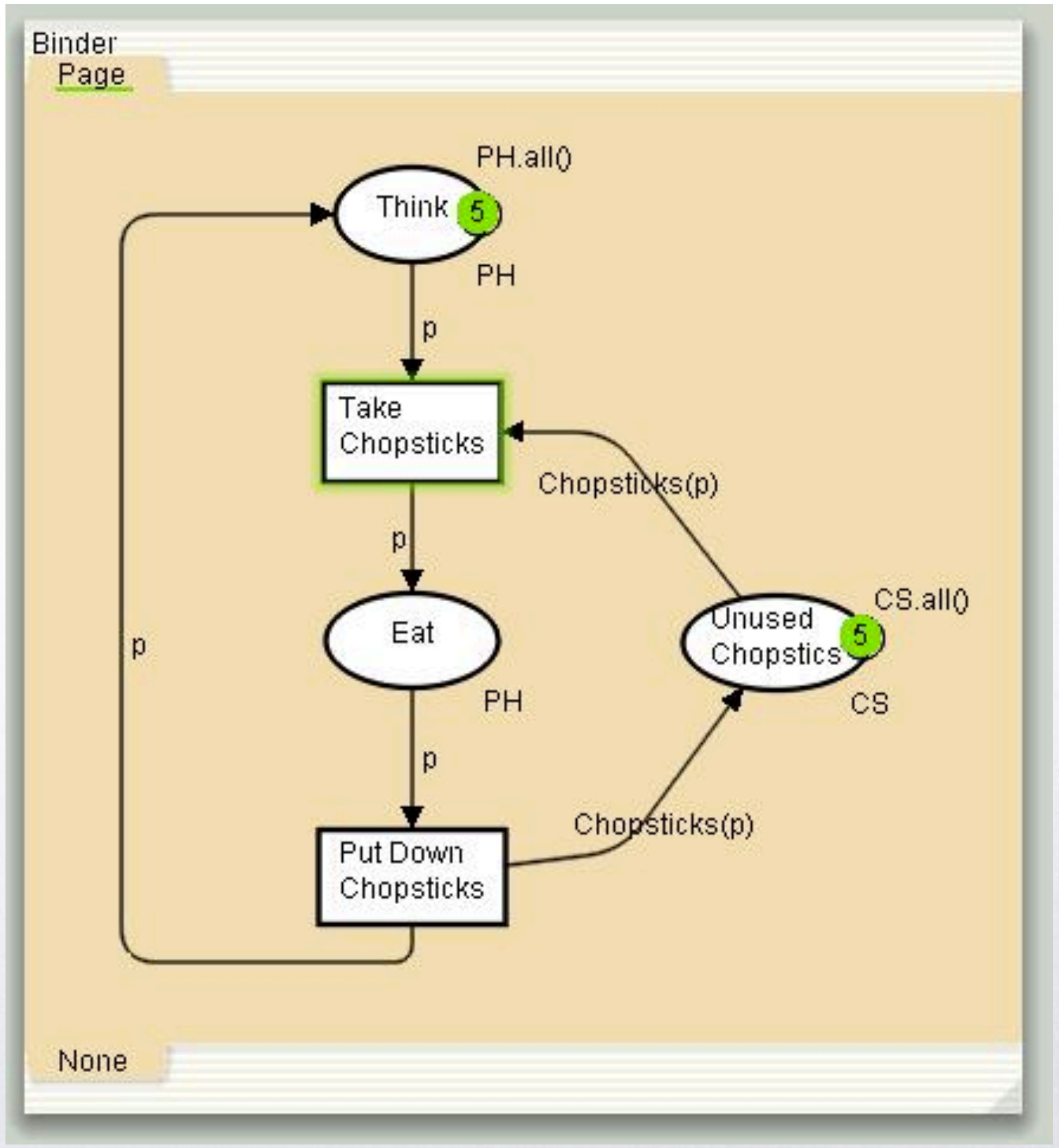


$P=\{p_1, p_2, p_3\}$
$H=\{h_1, h_2, h_3\}$
$U=P \cup H$

$l : P \rightarrow H$
$p_i \rightarrow h_i$
$r : P \rightarrow H$
$p_1 \rightarrow h_2$
$p_2 \rightarrow h_3$
$p_3 \rightarrow h_1$


Philosopher
P1
P2
P3
P4
P5
noodles

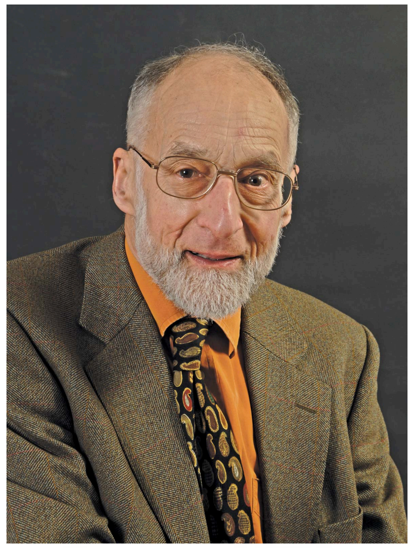

Dining Philosophers A


Binder
Page
PH.all()
Think
5
PH
p
Take Chopsticks
Chopsticks(p)
p
Eat
PH
Unused Chopstics
5
CS.all()
CS
p
p
Put Down Chopsticks
Chopsticks(p)
None

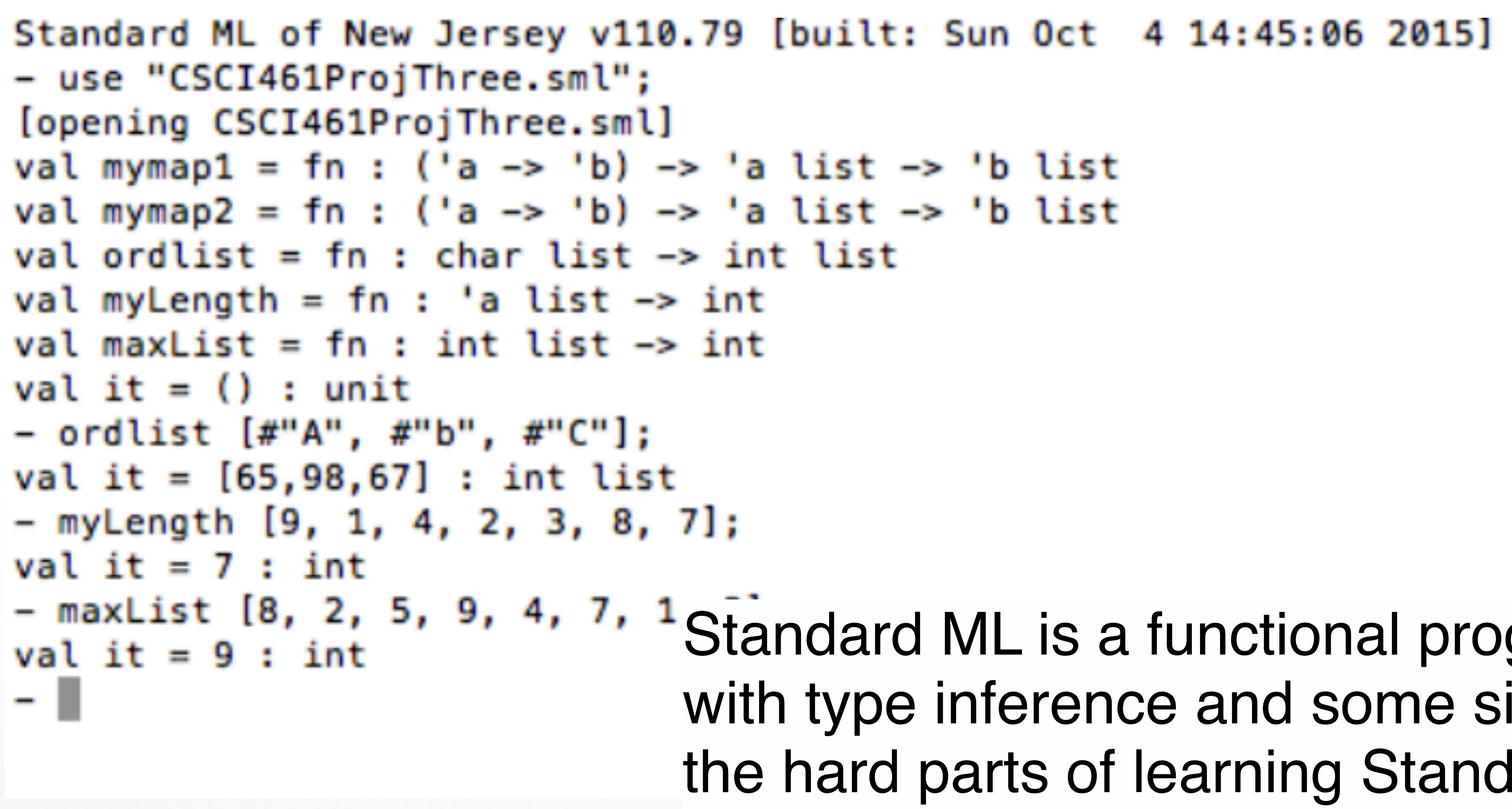
Informal CPN definition

Kurt Jensen

Coloured Petri Nets (CP-nets or CPNs) is a graphical language for constructing models of concurrent systems and analyzing their properties. CP-nets is a discrete-event modeling language combining Petri Nets and the functional programming language CPN ML which is based on Standard ML.


Robin Milner

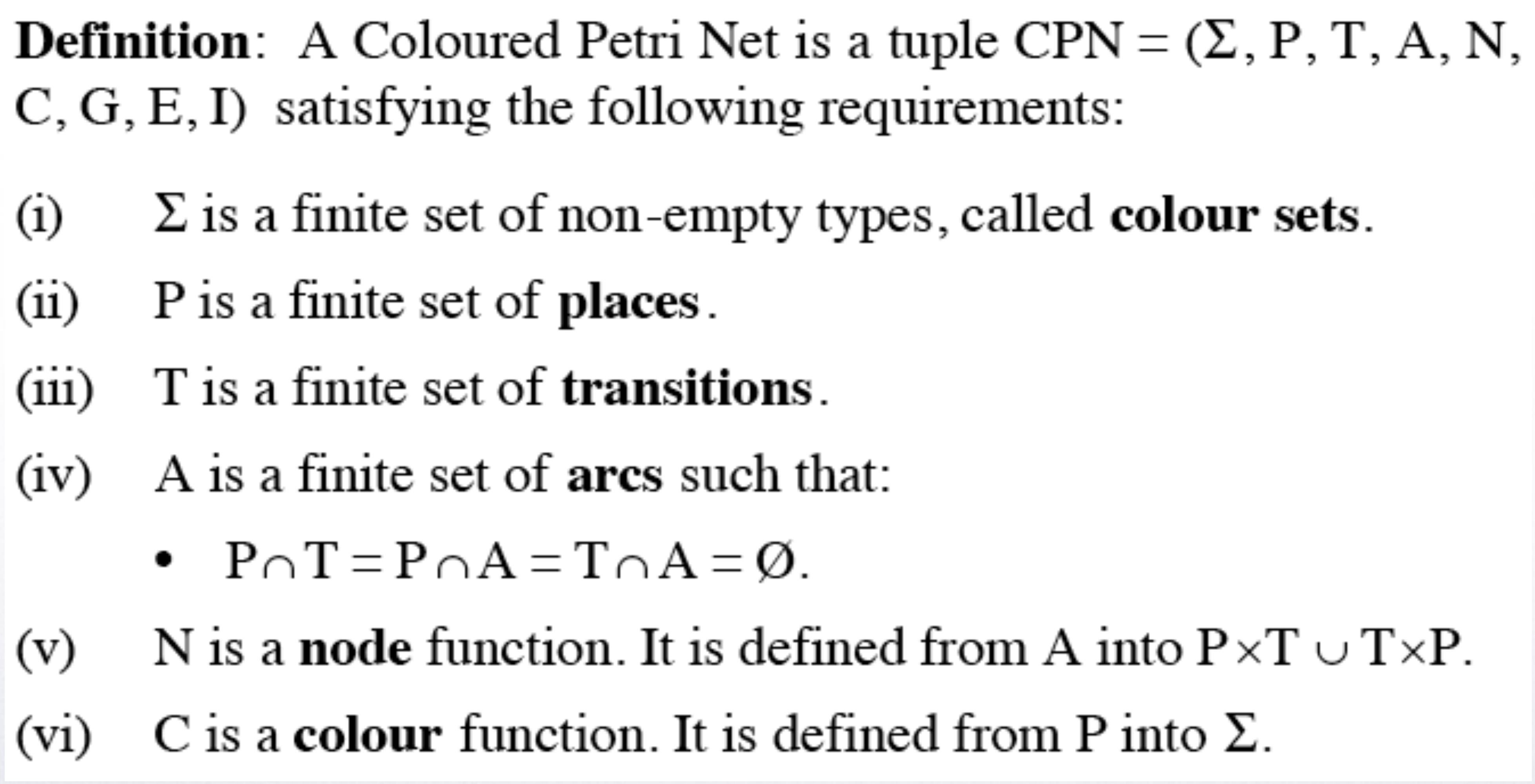
The ML family of strict functional languages, which includes F#, OCaml, and Standard ML, evolved from the *Meta Language* of the LCF theorem proving system developed by Robin Milner and his research group at the University of Edinburgh in the 1970s.



```
Standard ML of New Jersey v110.79 [built: Sun Oct  4 14:45:06 2015]
- use "CSCI461ProjThree.sml";
[opening CSCI461ProjThree.sml]
val mymap1 = fn : ('a -> 'b) -> 'a list -> 'b list
val mymap2 = fn : ('a -> 'b) -> 'a list -> 'b list
val ordlist = fn : char list -> int list
val myLength = fn : 'a list -> int
val maxList = fn : int list -> int
val it = () : unit
- ordlist [#"A", #"b", #"C"];
val it = [65,98,67] : int list
- myLength [9, 1, 4, 2, 3, 8, 7];
val it = 7 : int
- maxList [8, 2, 5, 9, 4, 7, 1
val it = 9 : int
- 
```

Standard ML is a functional programming language with type inference and some side-effects. Some of the hard parts of learning Standard ML are: Recursion, pattern matching, type inference (guessing the right types but never allowing implicit type conversion). Standard ML is distinguished from Haskell by including references, allowing variables to be updated.

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**.

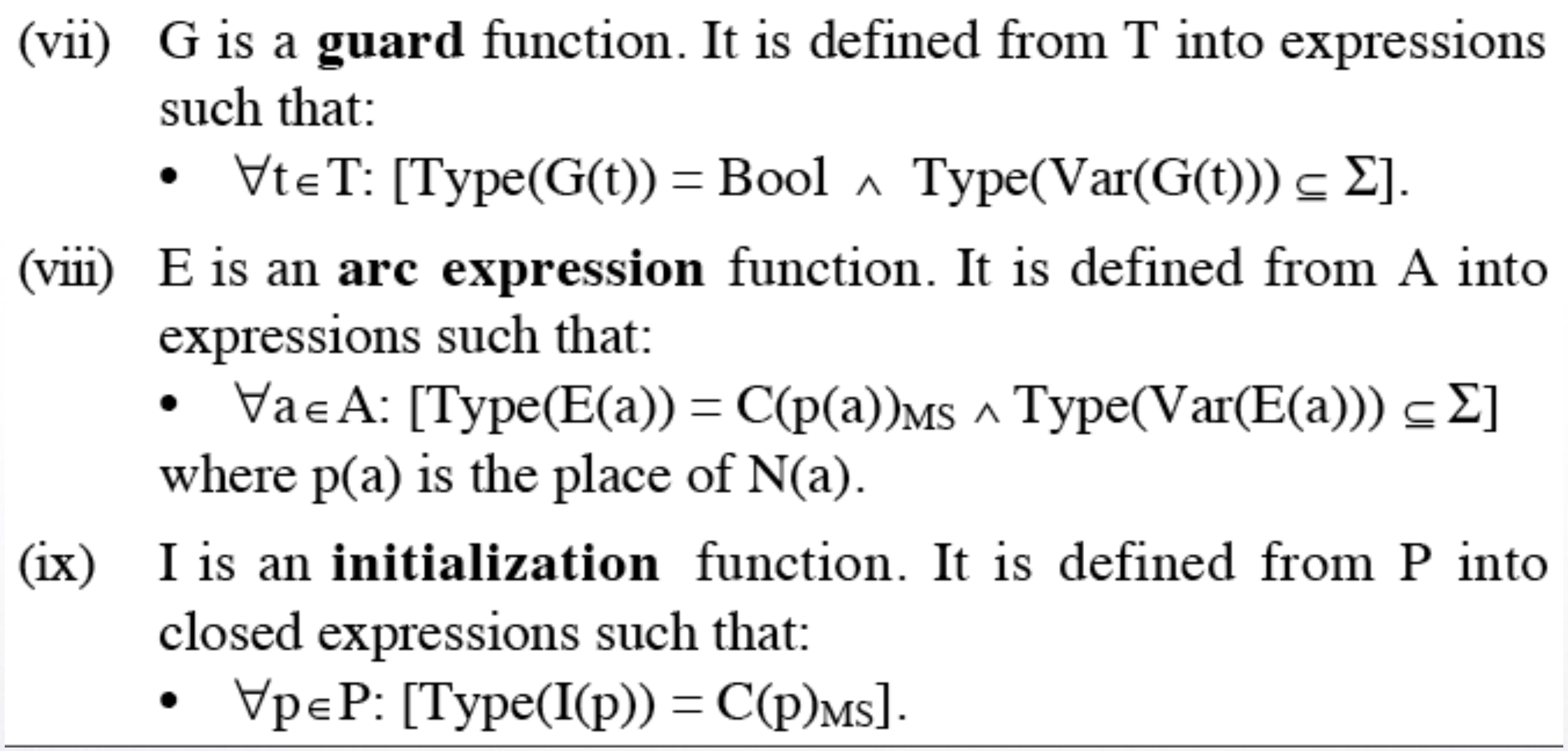
(iii) T is a finite set of **transitions**.

(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:

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

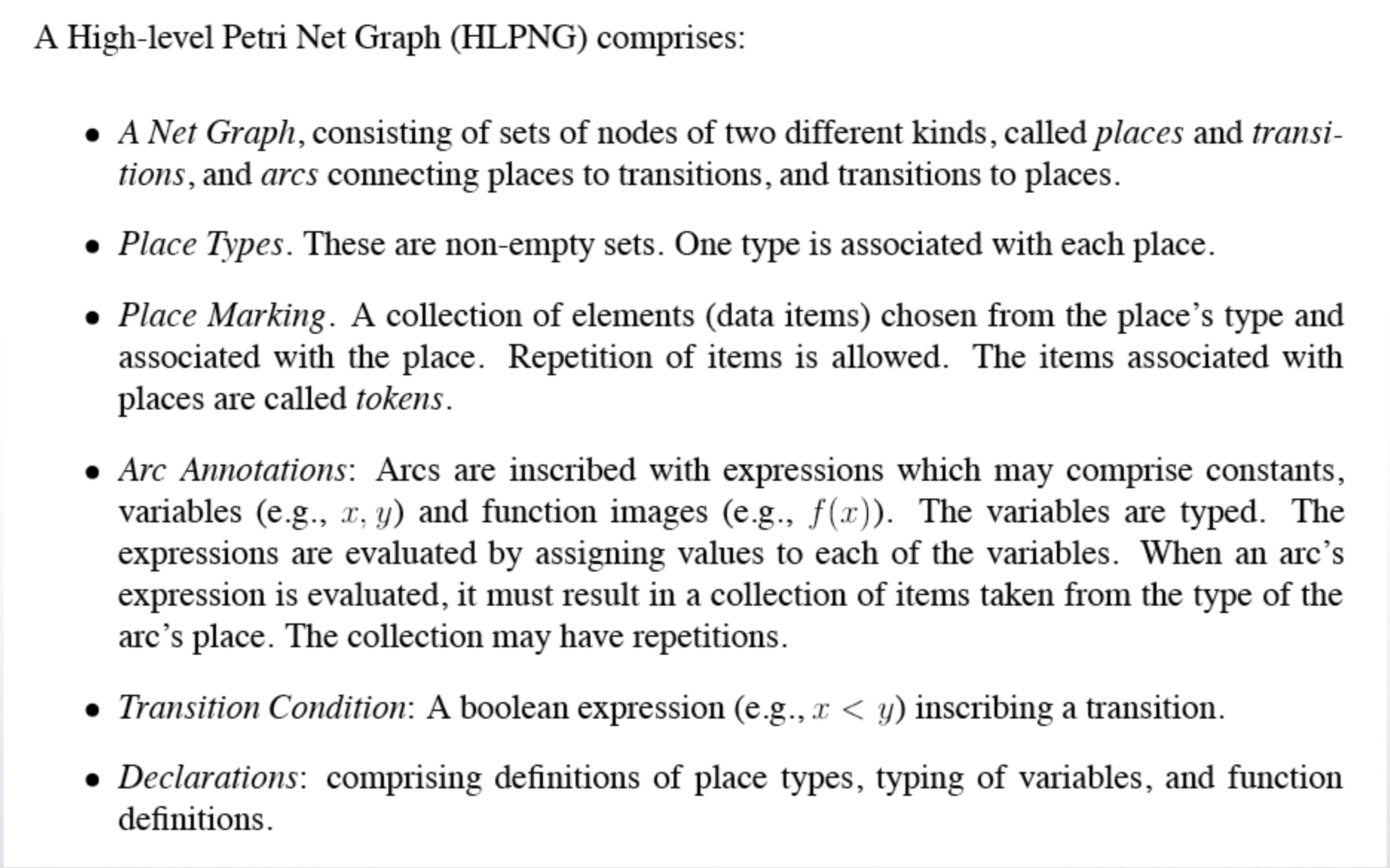
(viii) E is an **arc expression** function. It is defined from A into expressions such that:

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

where p(a) is the place of N(a).

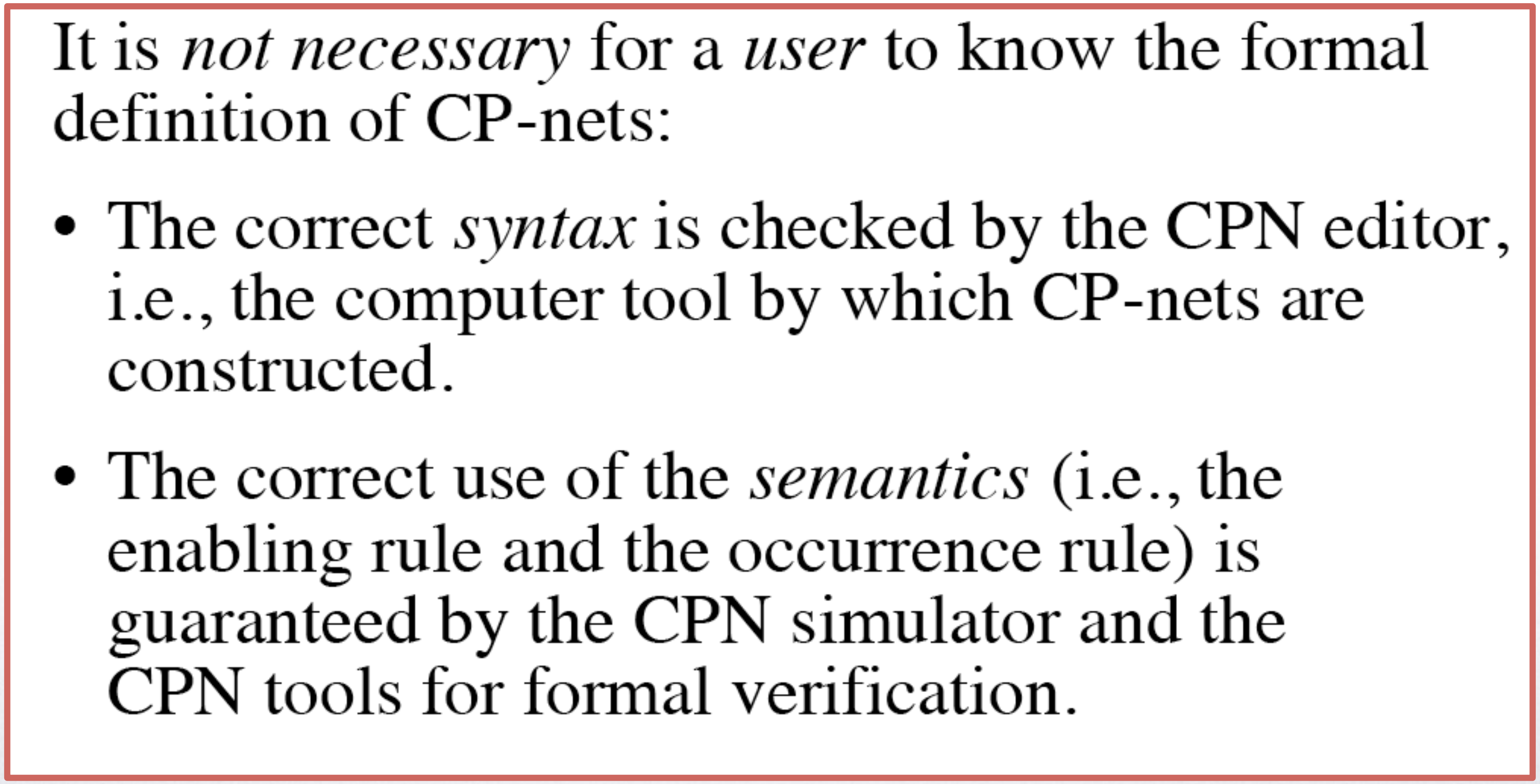
(ix) I is an **initialization** function. It is defined from P into closed expressions such that:

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



A High-level Petri Net Graph (HLPNG) comprises:

- *A Net Graph*, consisting of sets of nodes of two different kinds, called *places* and *transitions*, and *arcs* connecting places to transitions, and transitions to places.
- *Place Types*. These are non-empty sets. One type is associated with each place.
- *Place Marking*. A collection of elements (data items) chosen from the place's type and associated with the place. Repetition of items is allowed. The items associated with places are called *tokens*.
- *Arc Annotations*: Arcs are inscribed with expressions which may comprise constants, variables (e.g., x, y) and function images (e.g., $f(x)$). The variables are typed. The expressions are evaluated by assigning values to each of the variables. When an arc's expression is evaluated, it must result in a collection of items taken from the type of the arc's place. The collection may have repetitions.
- *Transition Condition*: A boolean expression (e.g., $x < y$) inscribing a transition.
- *Declarations*: comprising definitions of place types, typing of variables, and function definitions.


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).

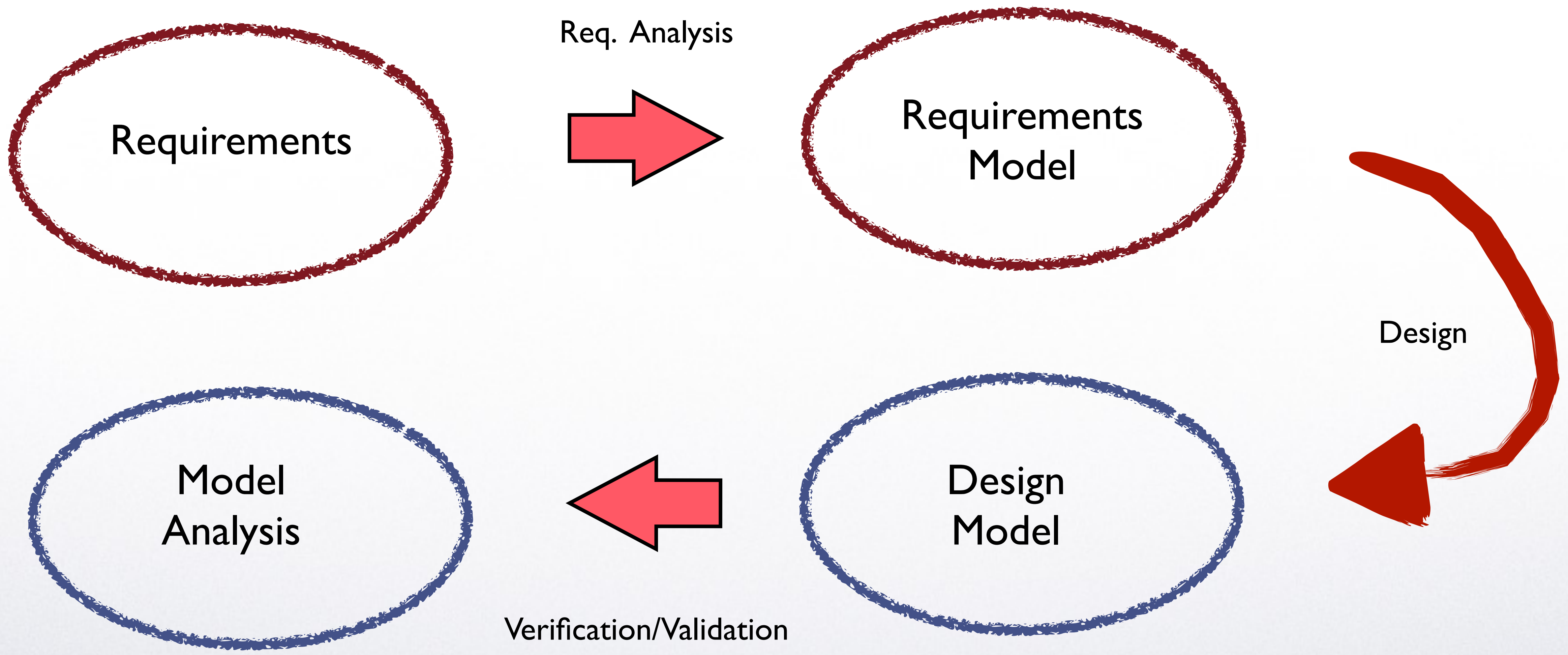
Even if environments can be an apprentice for that there is no guarantee to get a good metaphor. Only going through the modeling, analysis and verification we can be certain about "correctness".

Interpretations can also be :

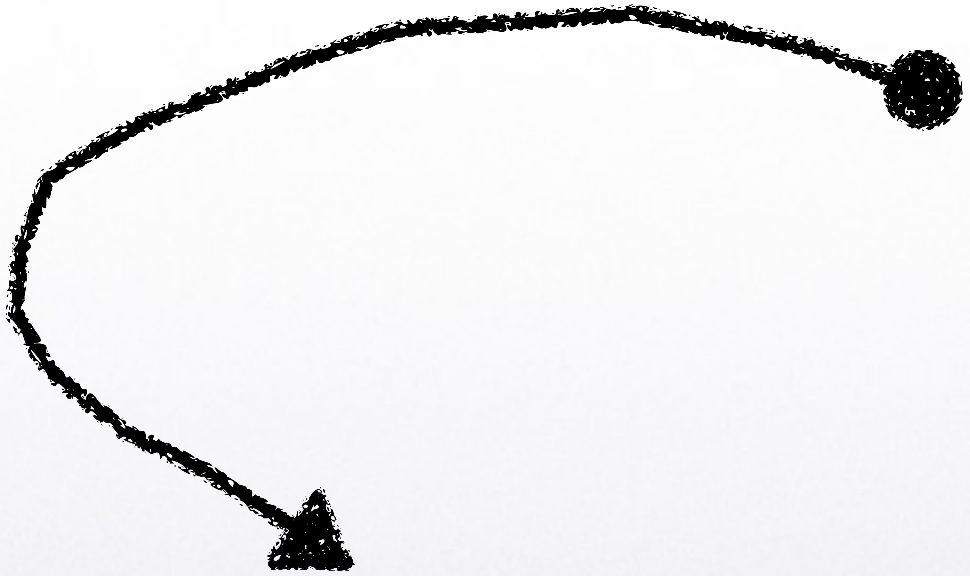
abstract (like in requirements analysis)

concrete (like in PLC programming)

Use of Petri Nets in Design



The gas station problem



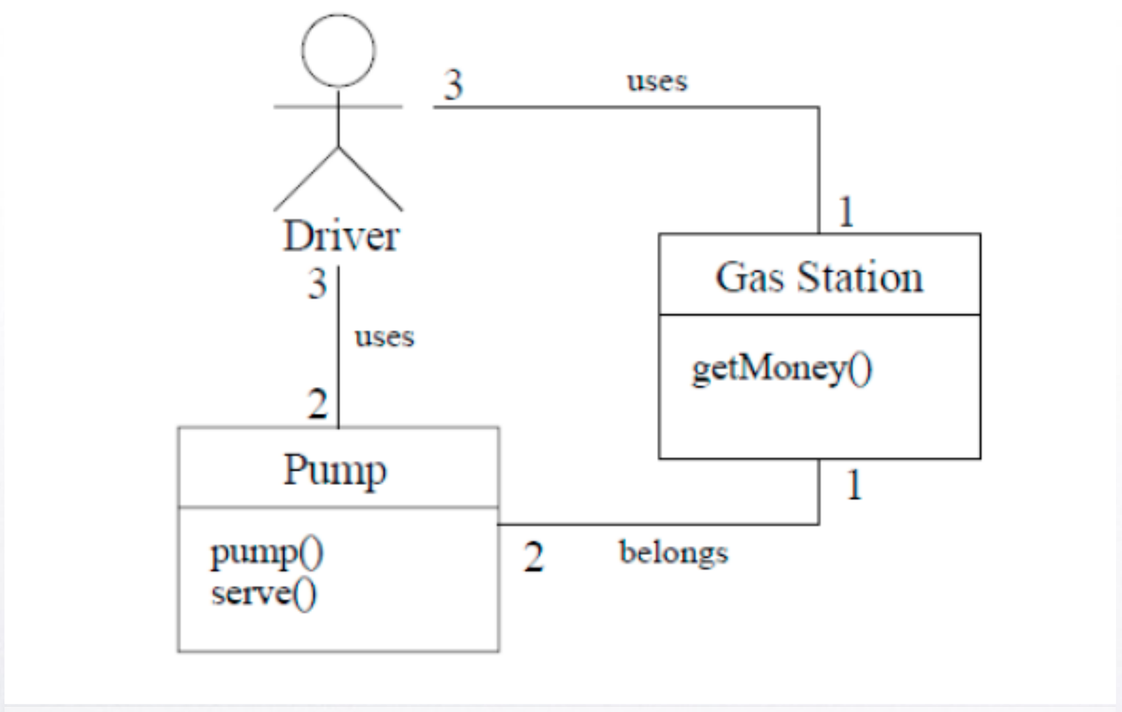
Exit

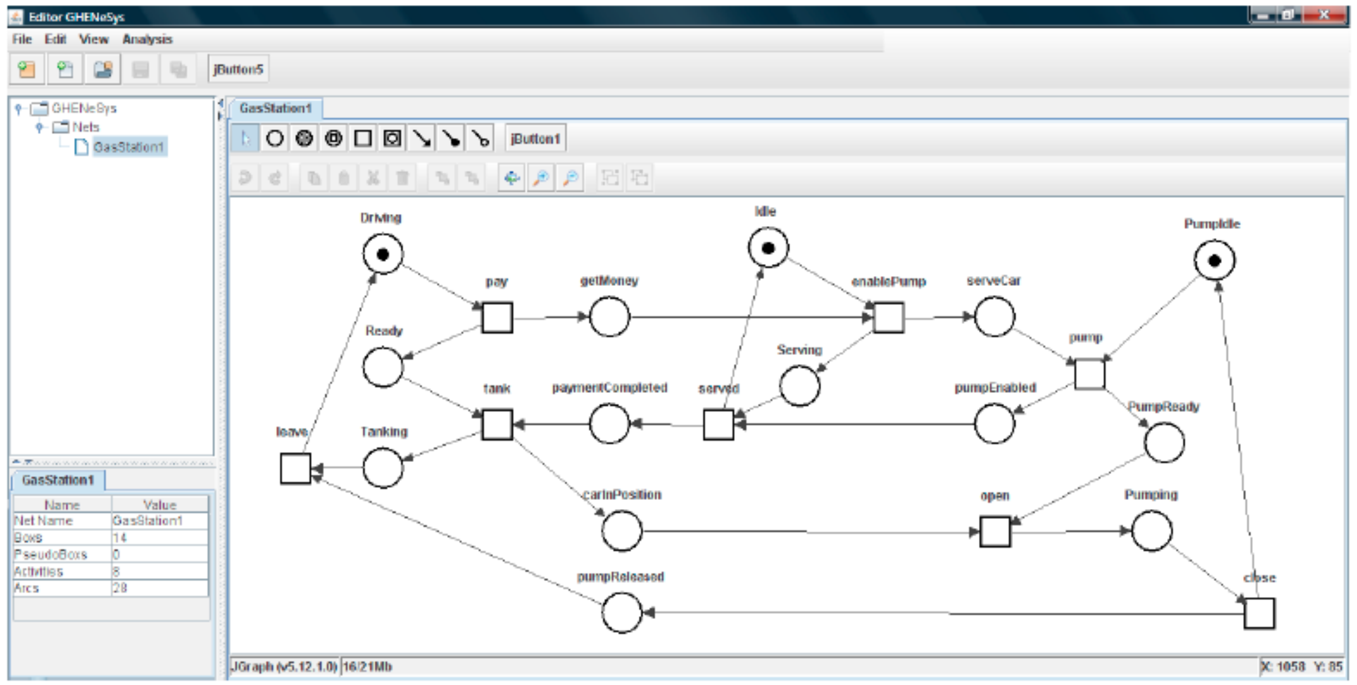
Queremos automatizar quase completamente (esta é uma abertura do projeto, ter alguém para receber o pagamento ou usar cartão somente) o serviço de atendimento em um posto de gasolina. Os sub-serviços seriam o abastecimento de combustível, gás, óleo (calibração seria gratuita). Precisamos portanto fazer o design do "work flow" para que os clientes não tenham que esperar em longas filas, e para que o posto não tenha prejuízo financeiro. É preciso distribuir os clientes nos serviços para obter este objetivo.

Será que é possível modelar este problema em redes CPN ao invés de seguir o caminho canônico da modelagem em redes P/T e o dobramento?

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.





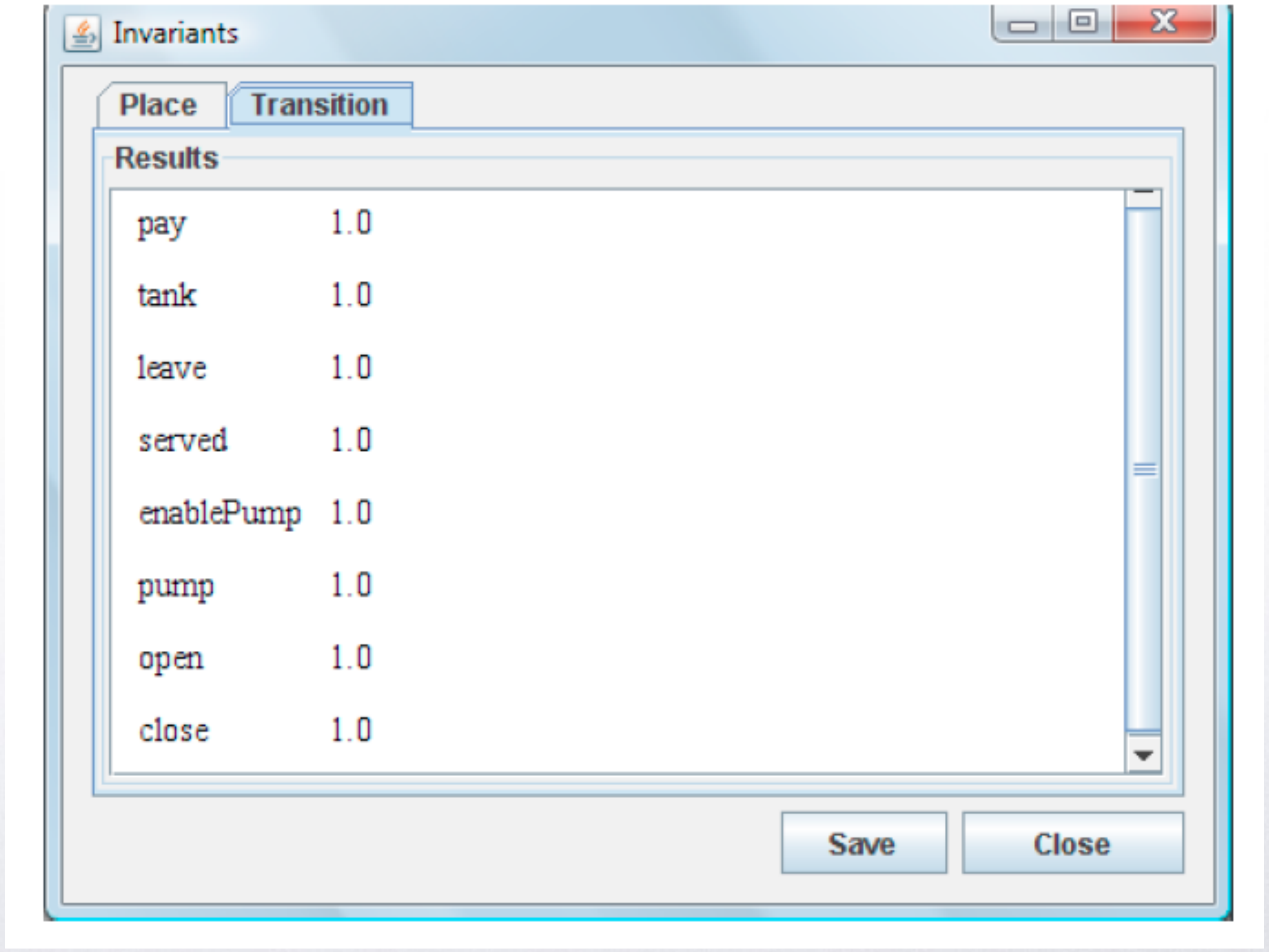
GasStation1

Name	Value
Net Name	GasStation1
Boxs	14
PseudoBoxs	0
Activities	8
Arcs	28

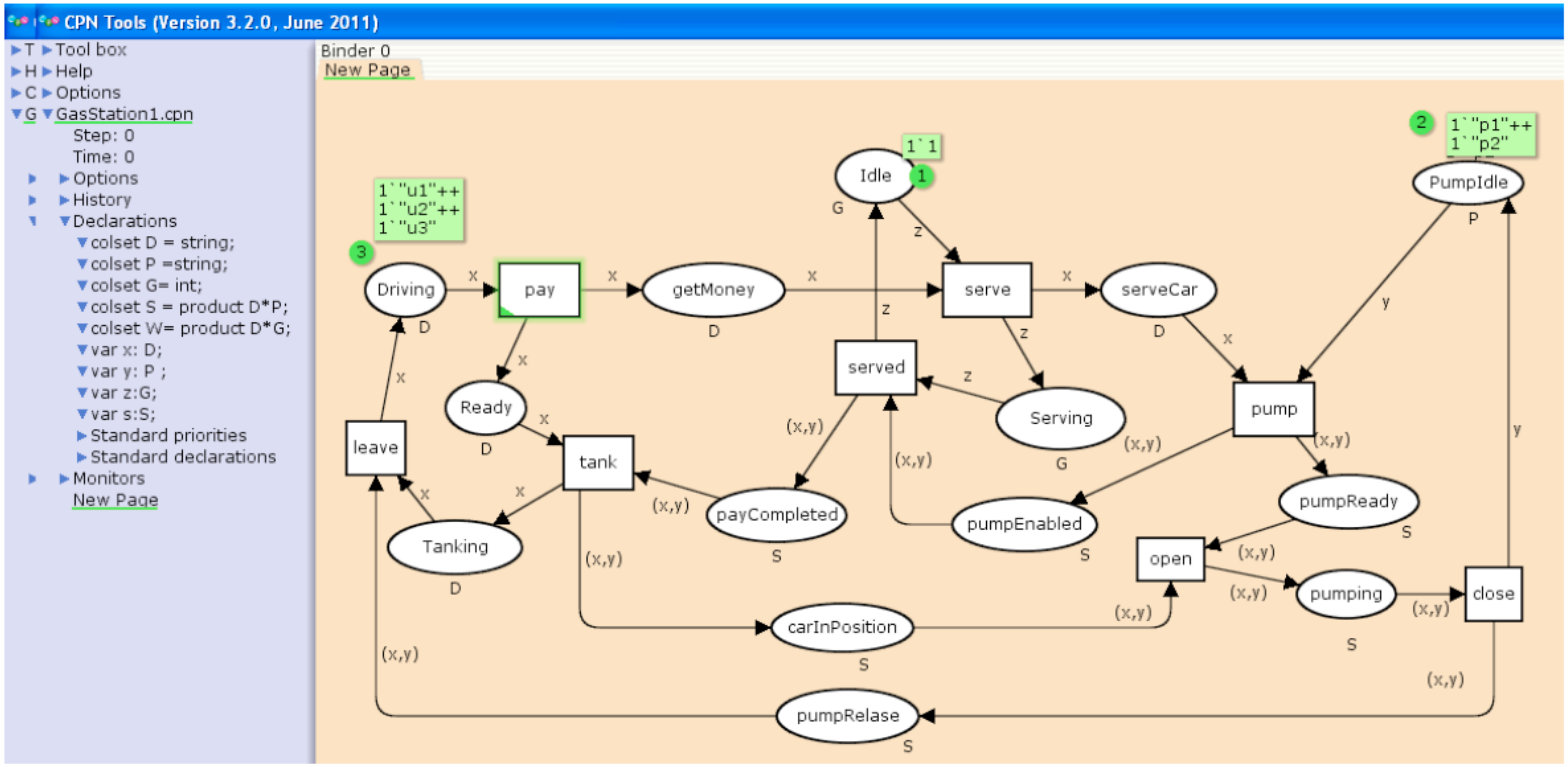
Baresi, L. and Pezze, M., 2001. "Improving UML with petri nets". In *UNIGRA 2001, Uniform Approaches to Graphical Process Specification Techniques (a Satellite Event of ETAPS 2001)*. Elsevier, Vol. 44 of *Electronic Notes in Theoretical Computer Science*, pp. 107–119.

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

CPN Analysis

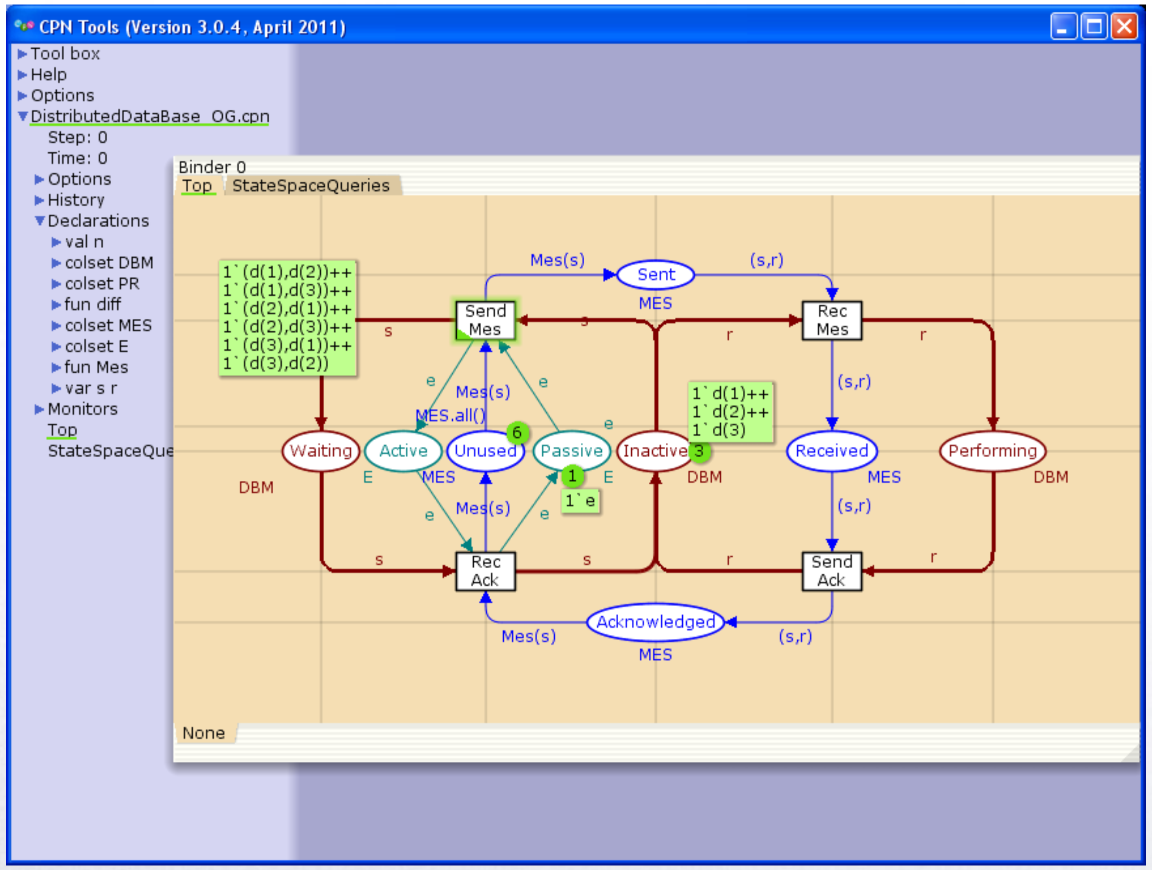
CPN Analysis follow the same procedures than the classical analysis and face the same problems, even if has new formal resources to include.

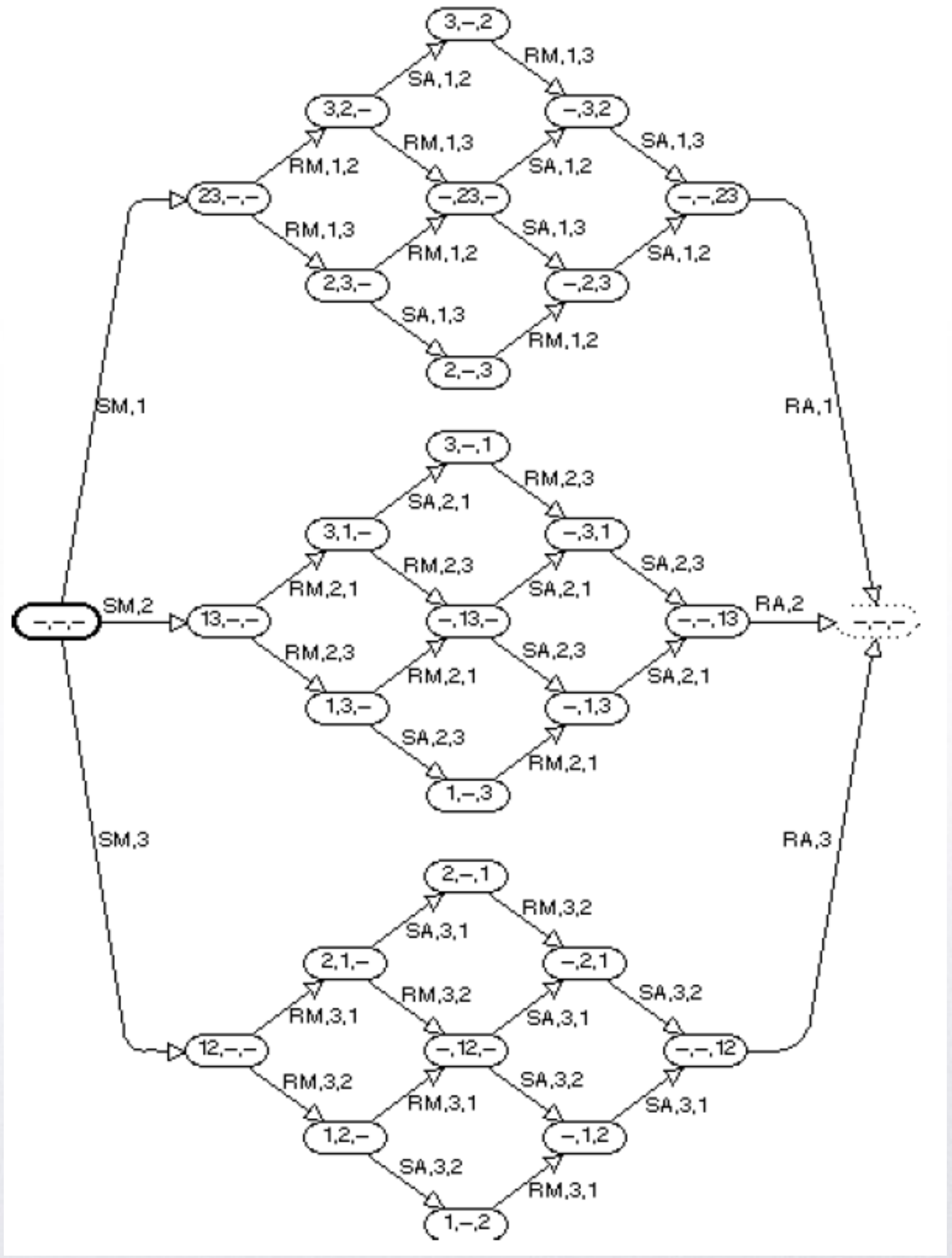
Again, we have a state/transition approach, in a discrete flavor, with the possibility of explosion of the number of combinations of states, that is, in the composition of processes.

Classic and HL modelling

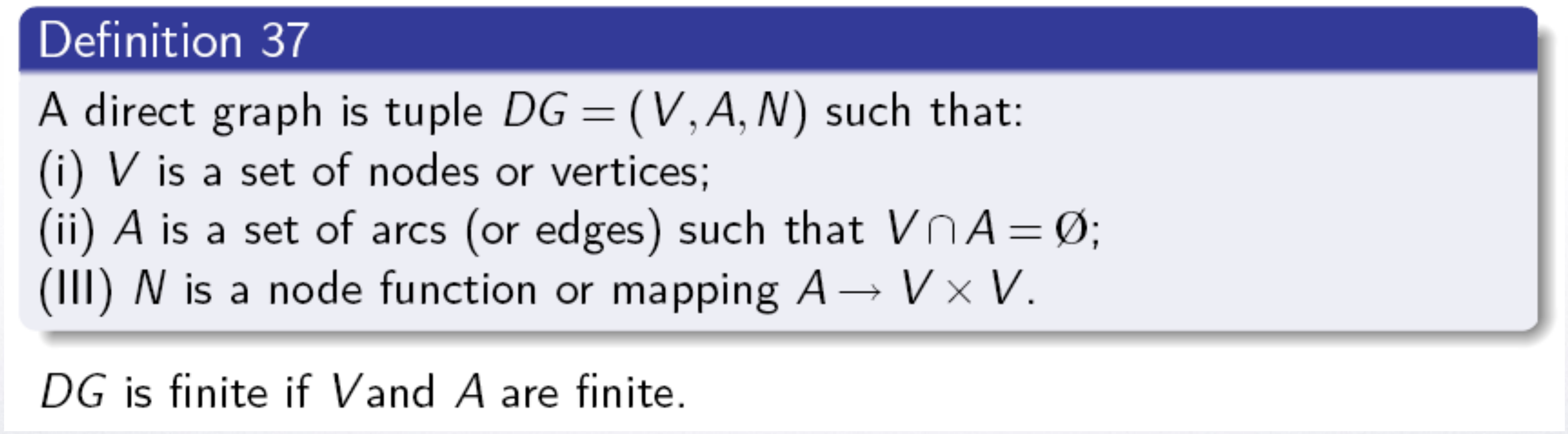
In fact, modeling in Classic P/T nets is very similar to modeling in CPN or other High Level net. However, the inclusion of type theory and consequently distinguished marking should not be underestimated, specially to model complex systems. This is the feature that open the possibility to apply Petri nets to more (and different domais.

Analysis in CPN Nets





Directed Graphs

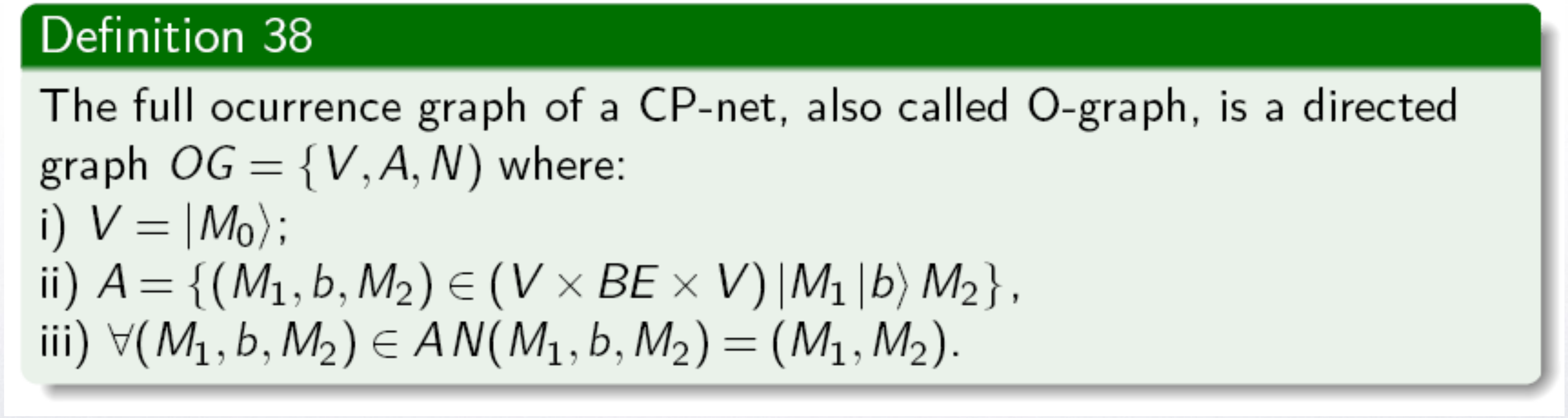


Definition 37

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

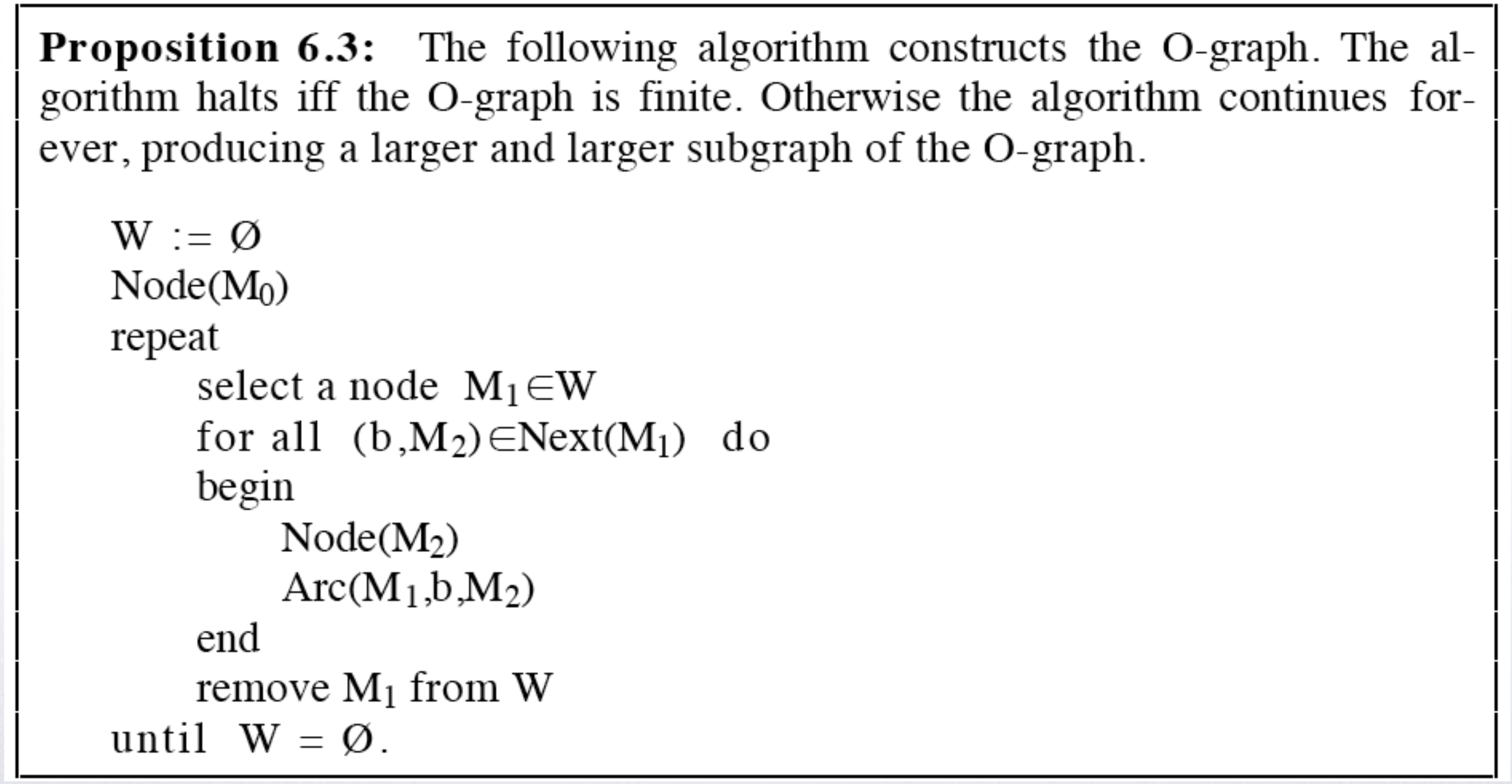
The full ocurrence graph of a CP-net, also called O-graph, is a directed graph $OG = \{V, A, N)$ where:

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) $\forall (M_1, b, M_2) \in A\, N(M_1, b, M_2) = (M_1, M_2)$.

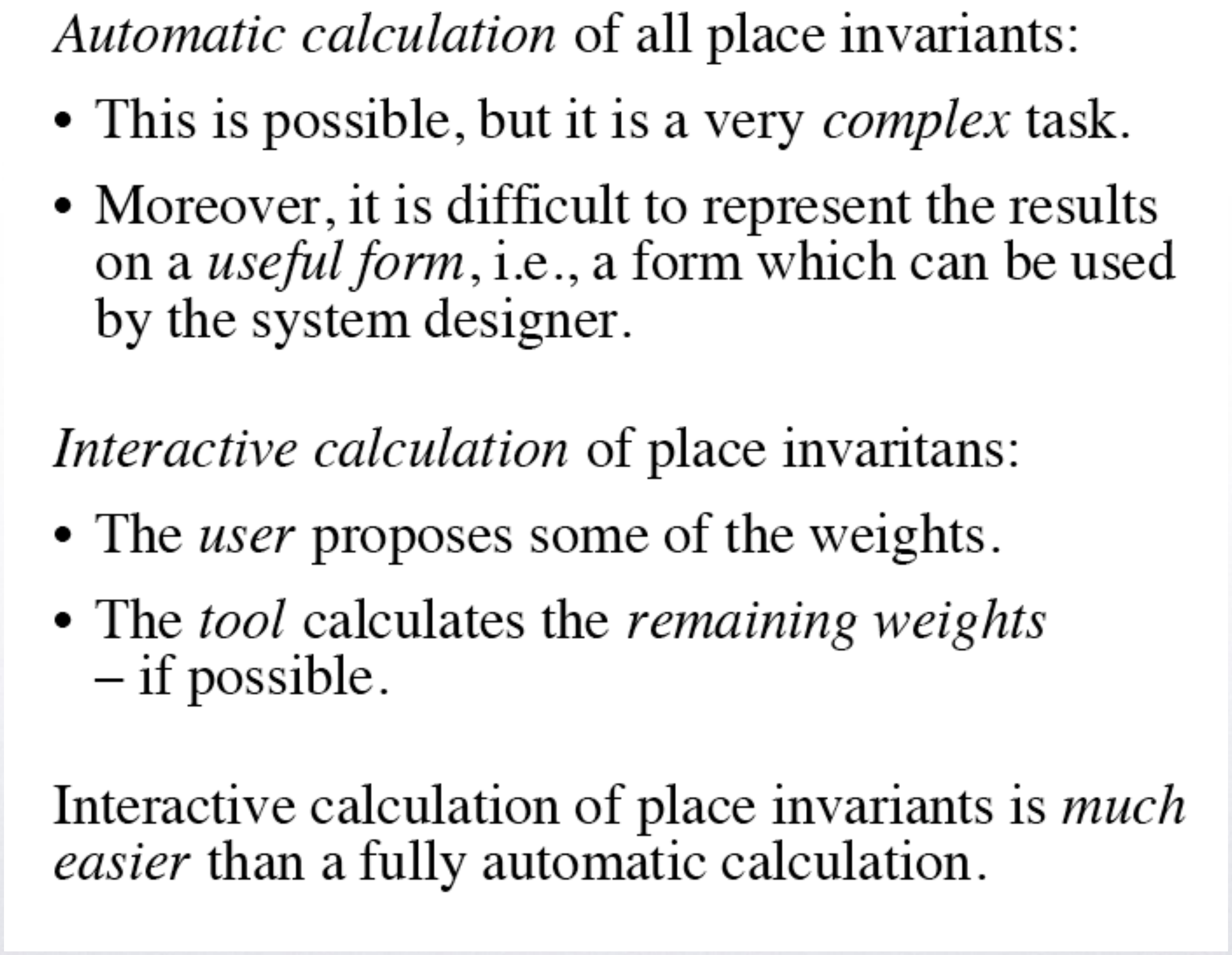
O Graph algorithm



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.
```

Automating the invariant analysis



Automatic calculation of all place invariants:

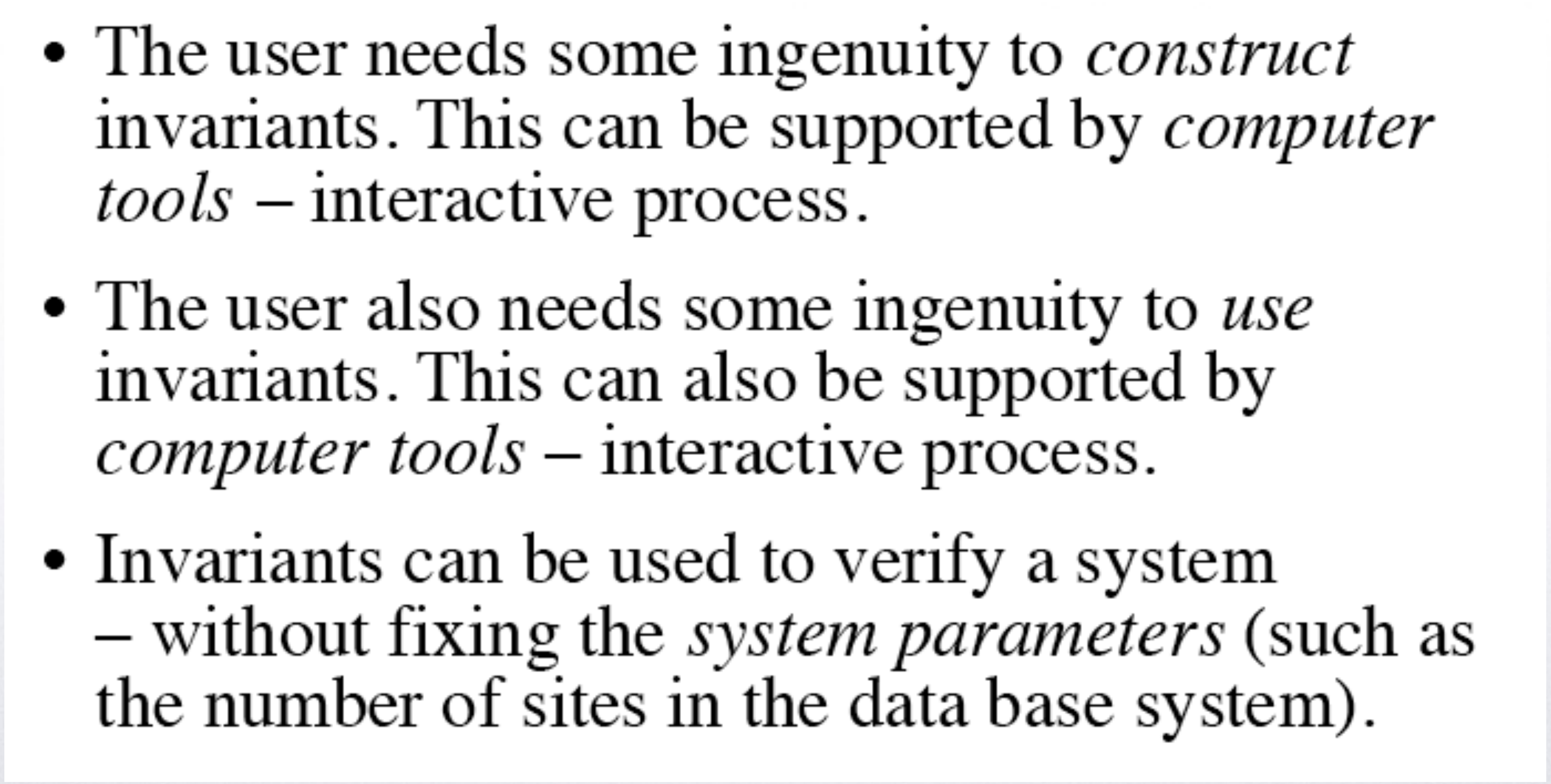
- This is possible, but it is a very *complex* task.
- Moreover, it is difficult to represent the results on a *useful form*, i.e., a form which can be used by the system designer.

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.

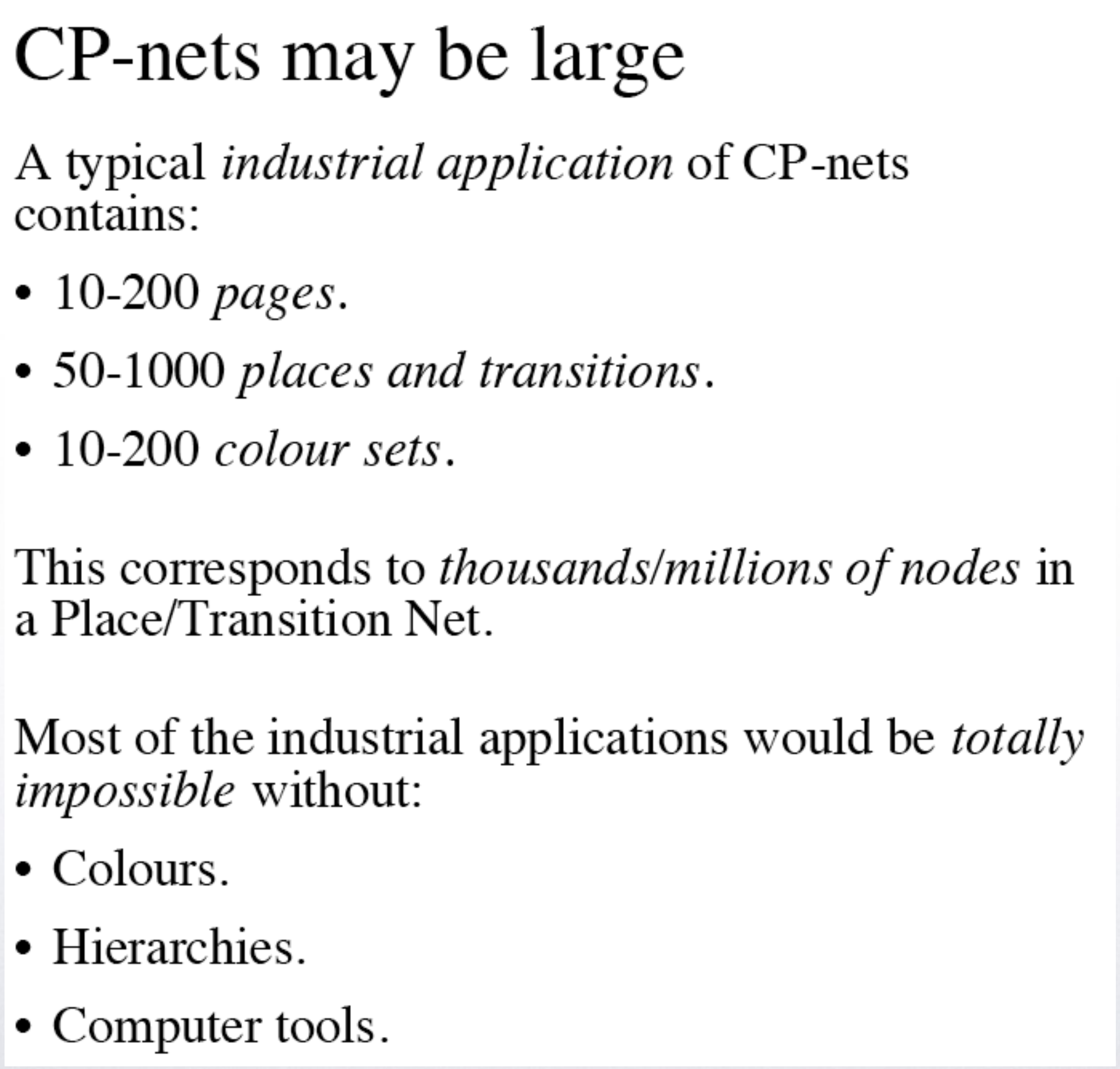
The invariant method in CPN



- 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:

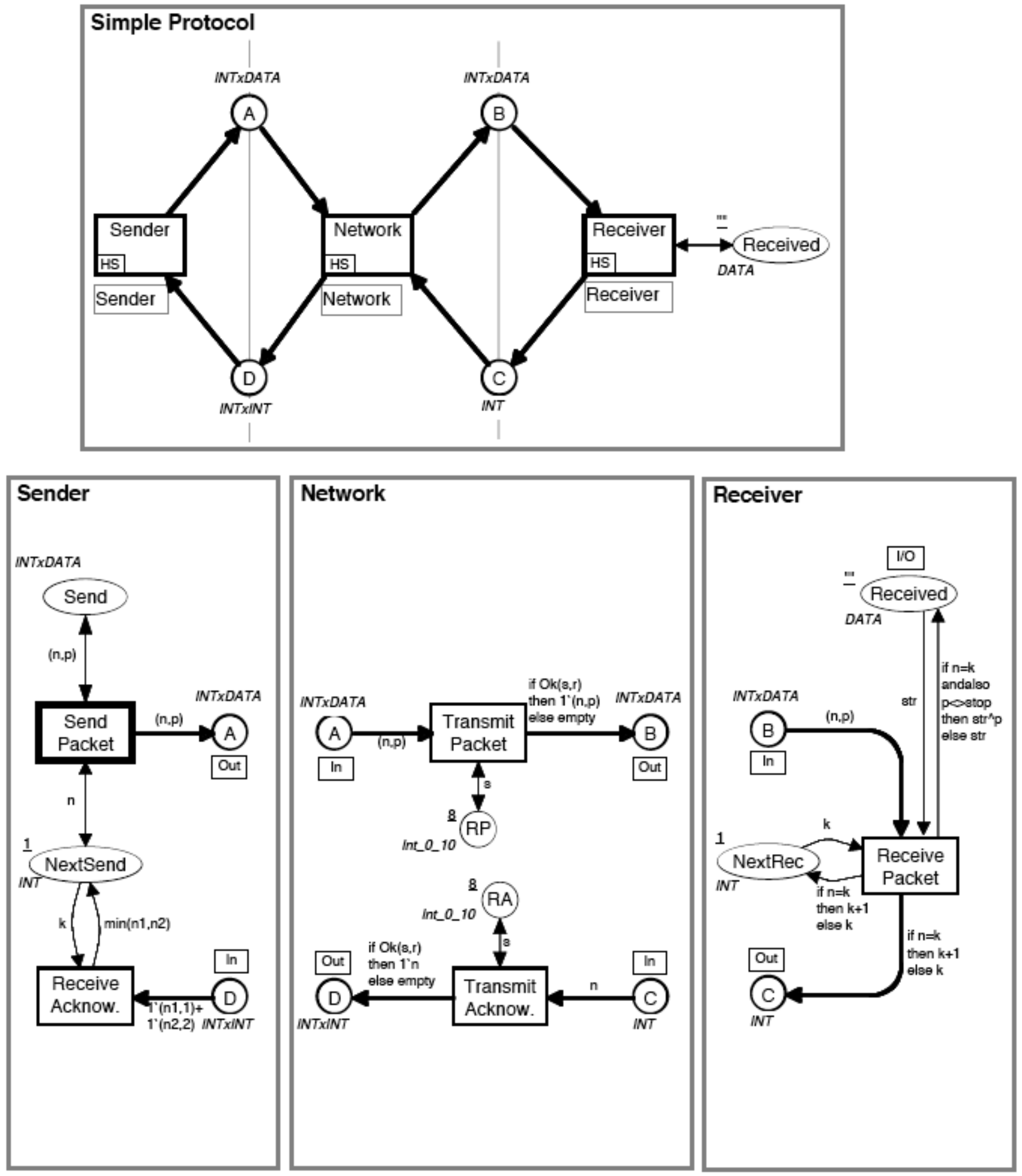
- 10-200 *pages*.
- 50-1000 *places and transitions*.
- 10-200 *colour sets*.

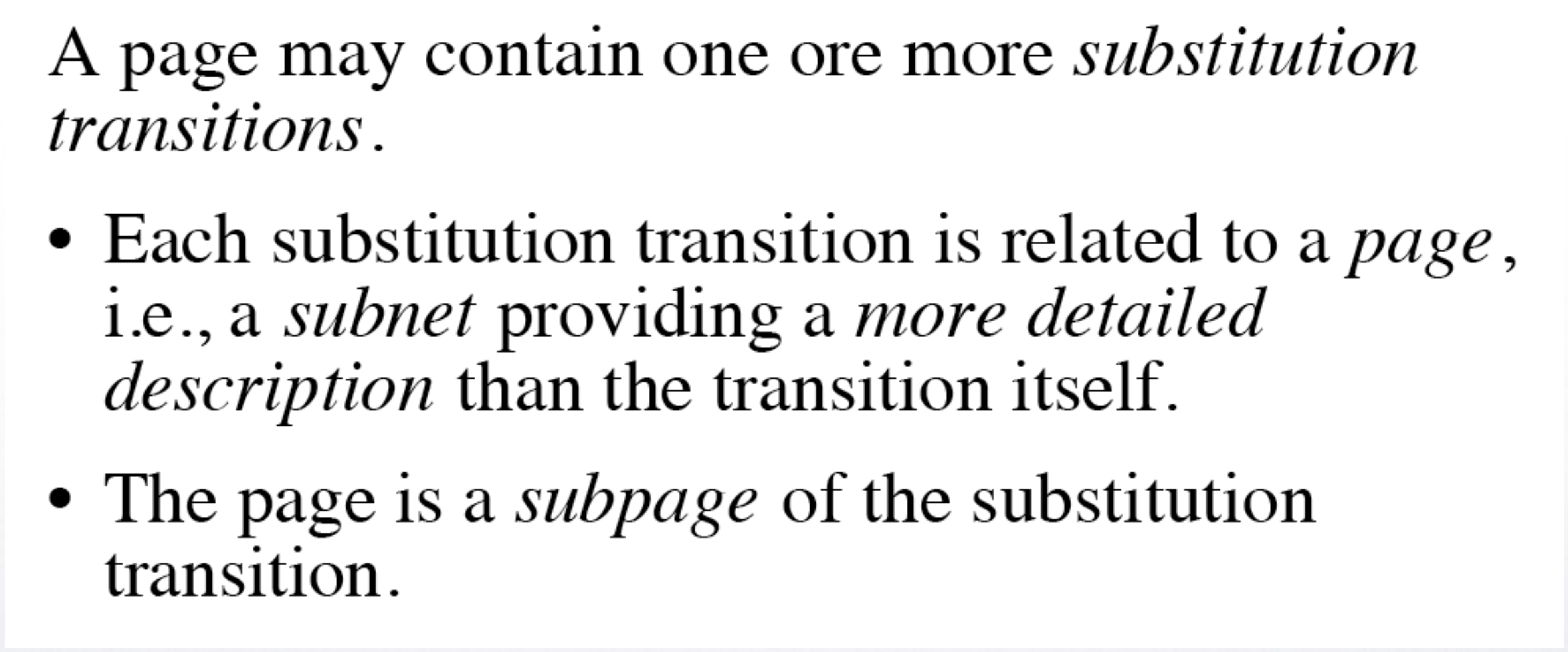
This corresponds to *thousands/millions of nodes* in a Place/Transition Net.

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*.



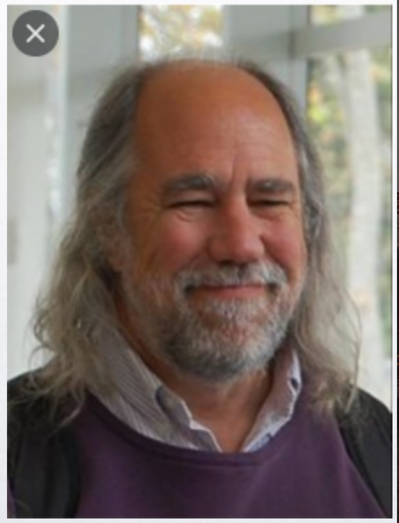


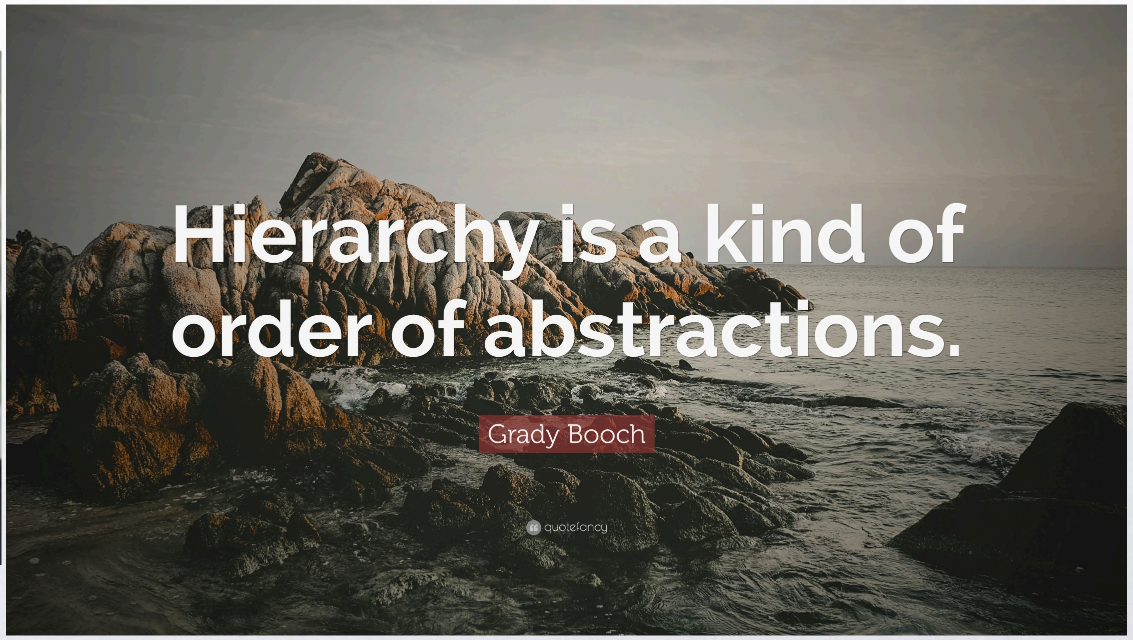
A page may contain one ore more *substitution transitions*.

- 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.

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

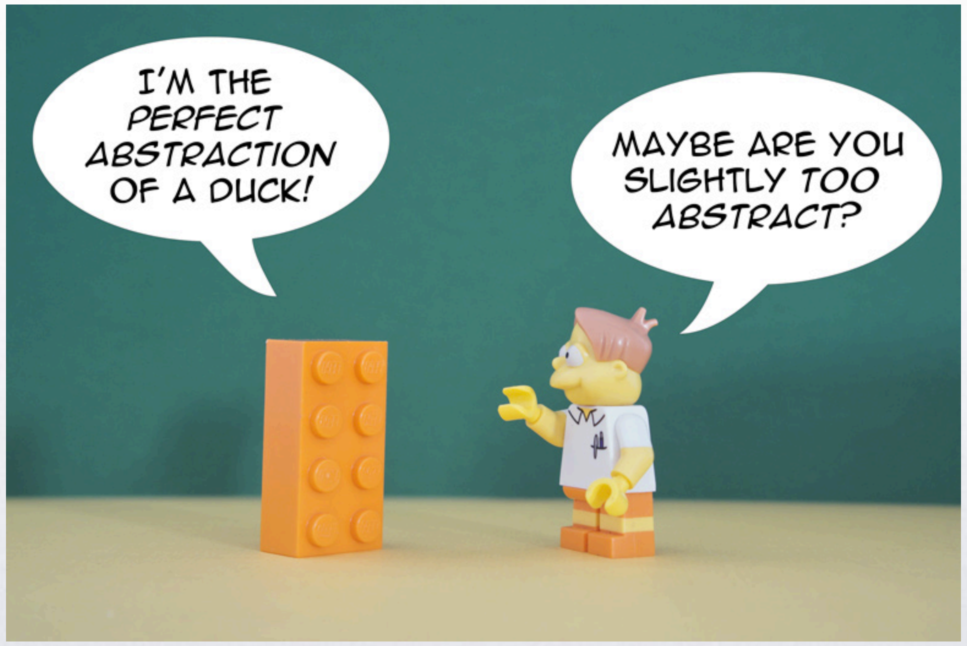
In design, hierarchy means to abstract the elements which properties are not relevant in an analysis phases.



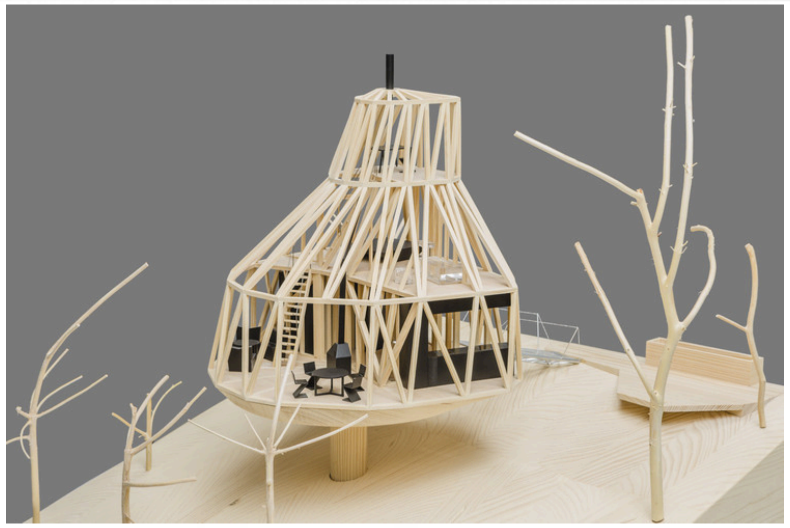


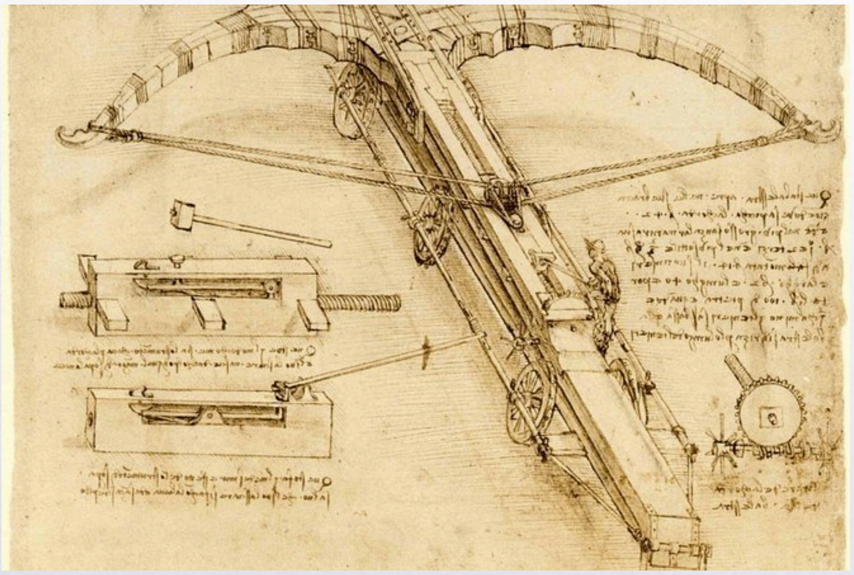
Hierarchy is a good abstraction feature. However, the real challenge is to associate that with the property analysis, so that the abstract net preserve the same properties than the expanded one.

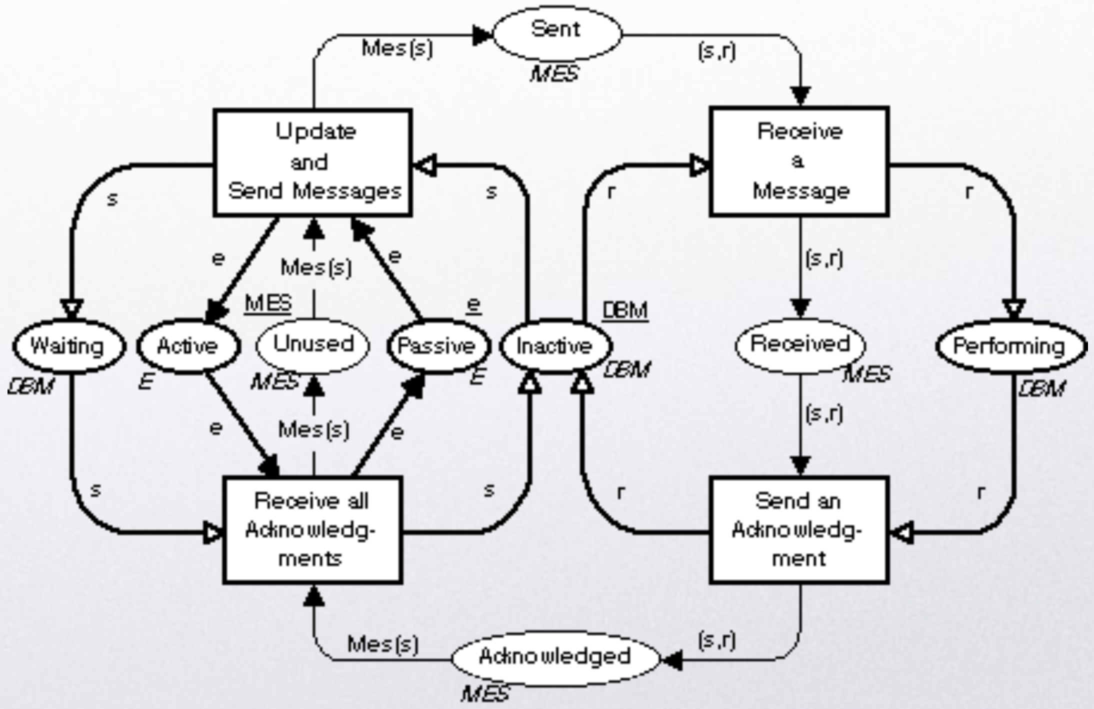
The proper requirement is a key issue for that.


I'M THE PERFECT ABSTRACTION OF A DUCK!
MAYBE ARE YOU SLIGHTLY TOO ABSTRACT?

Seja um modelo físico (mockup), uma descrição, ou um modelo teórico (hierárquico) o modelo e as partes abstratas devem refletir e preservar as propriedades atribuídas ao sistema físico em análise.







Em resumo...

Redes clássicas (P/T)

- Modelagem estado/transição
- Base teórica: grafos, álgebra linear

- Métodos de análise: geração da árvore de atingibilidade;
- Análise de propriedades <u>sobre a equação de estado</u>

Redes coloridas (CPN)

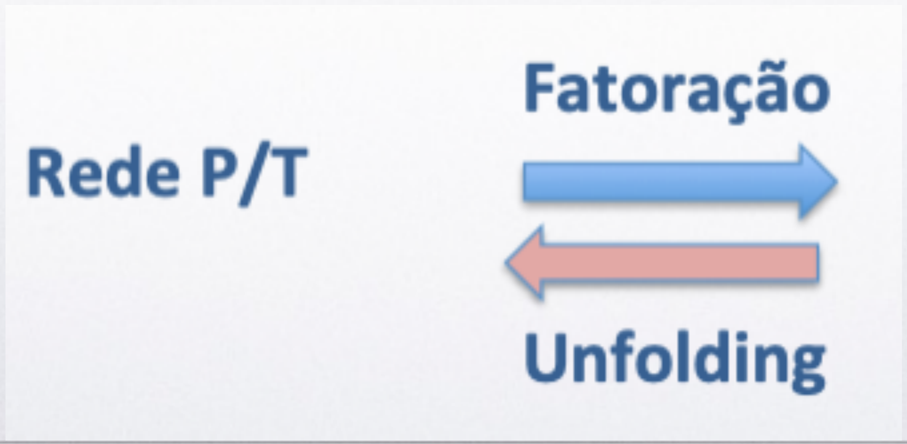
- Modelagem estado/transição
- Base teórica: grafos, Standard ML

- Métodos de análise: geração do O-graph;
- Análise de propriedades

Redes clássicas (P/T)

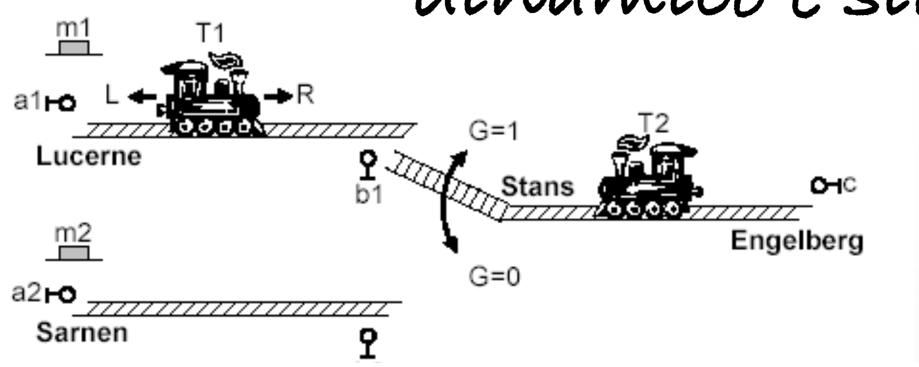
Redes coloridas (CPN)

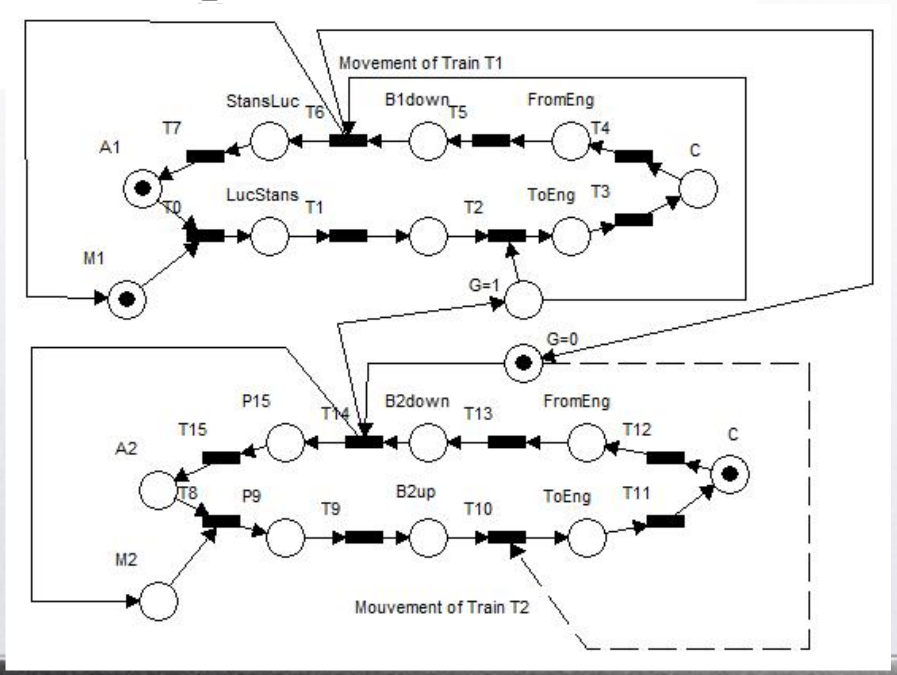
Mesma capacidade de expressão

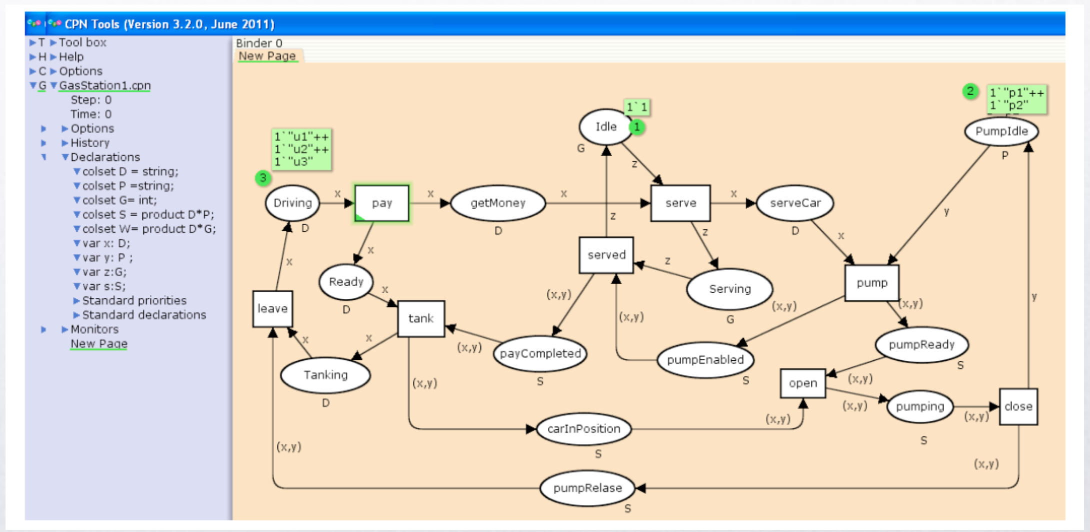


Rede de Alto Nível ou Colorida

Em ambos os casos o processo de modelagem de sistemas pode ser feito por abstração de um modelo físico ou arquetípico (que ainda não existe, ou requisitos), cujo comportamento dinâmico é sintetizado em uma rede.



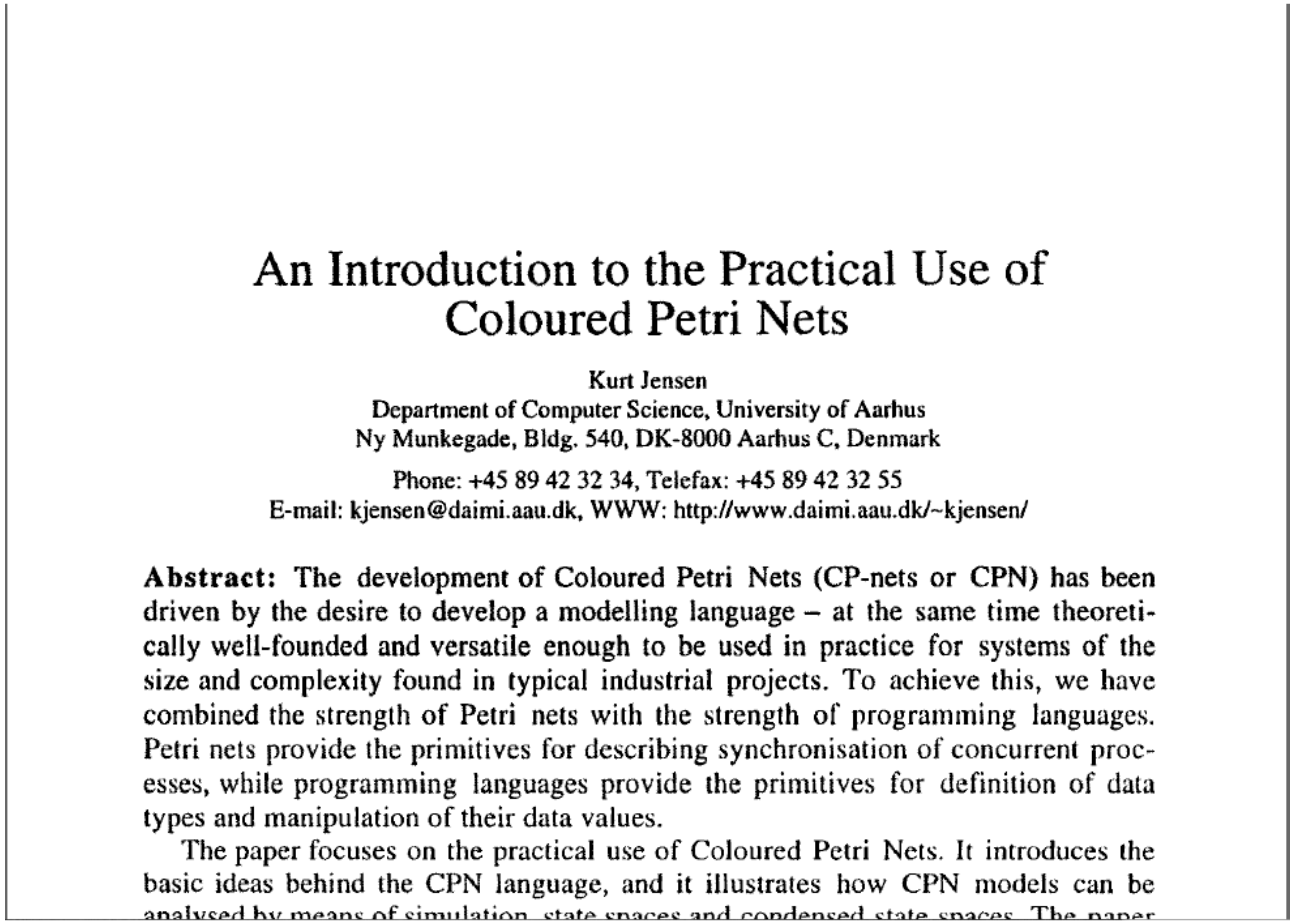



CPN Tools (Version 3.2.0, June 2011)

```
▶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
```

A diferença entre estes dois casos está na habilidade e intuição para sintetizar uma interpretação de um sistema físico (ou arquétipo) em uma rede P/T ou CPN. Certamente os modelos lineares e menos abstratos parecem mais simples.

Kurt Jensen
Univ. of Ahrus, Demark



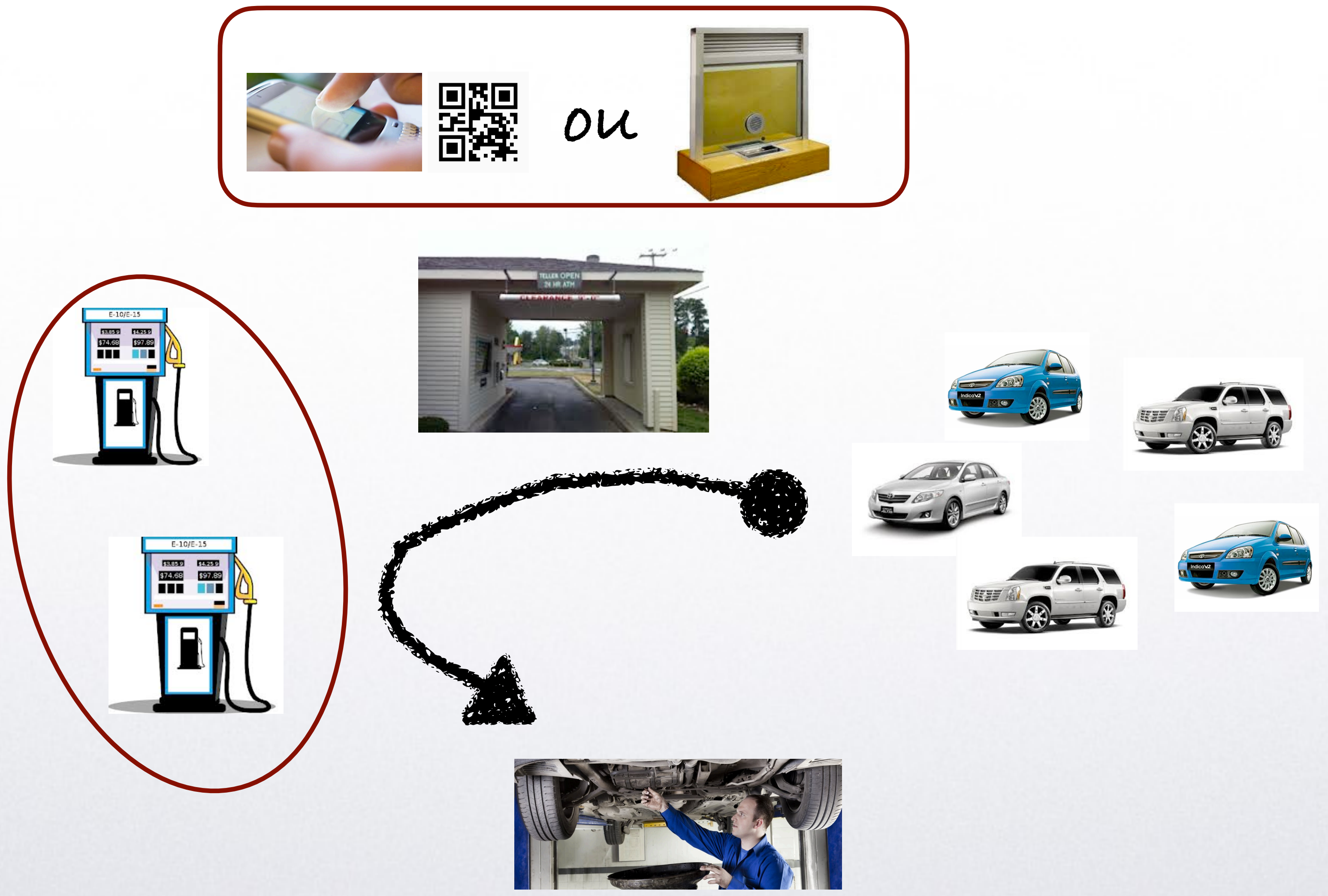
An Introduction to the Practical Use of Coloured Petri Nets

Kurt Jensen
Department of Computer Science, University of Aarhus
Ny Munkegade, Bldg. 540, DK-8000 Aarhus C, Denmark
Phone: +45 89 42 32 34, Telefax: +45 89 42 32 55
E-mail: kjensen@daimi.aau.dk, WWW: http://www.daimi.aau.dk/~kjensen/

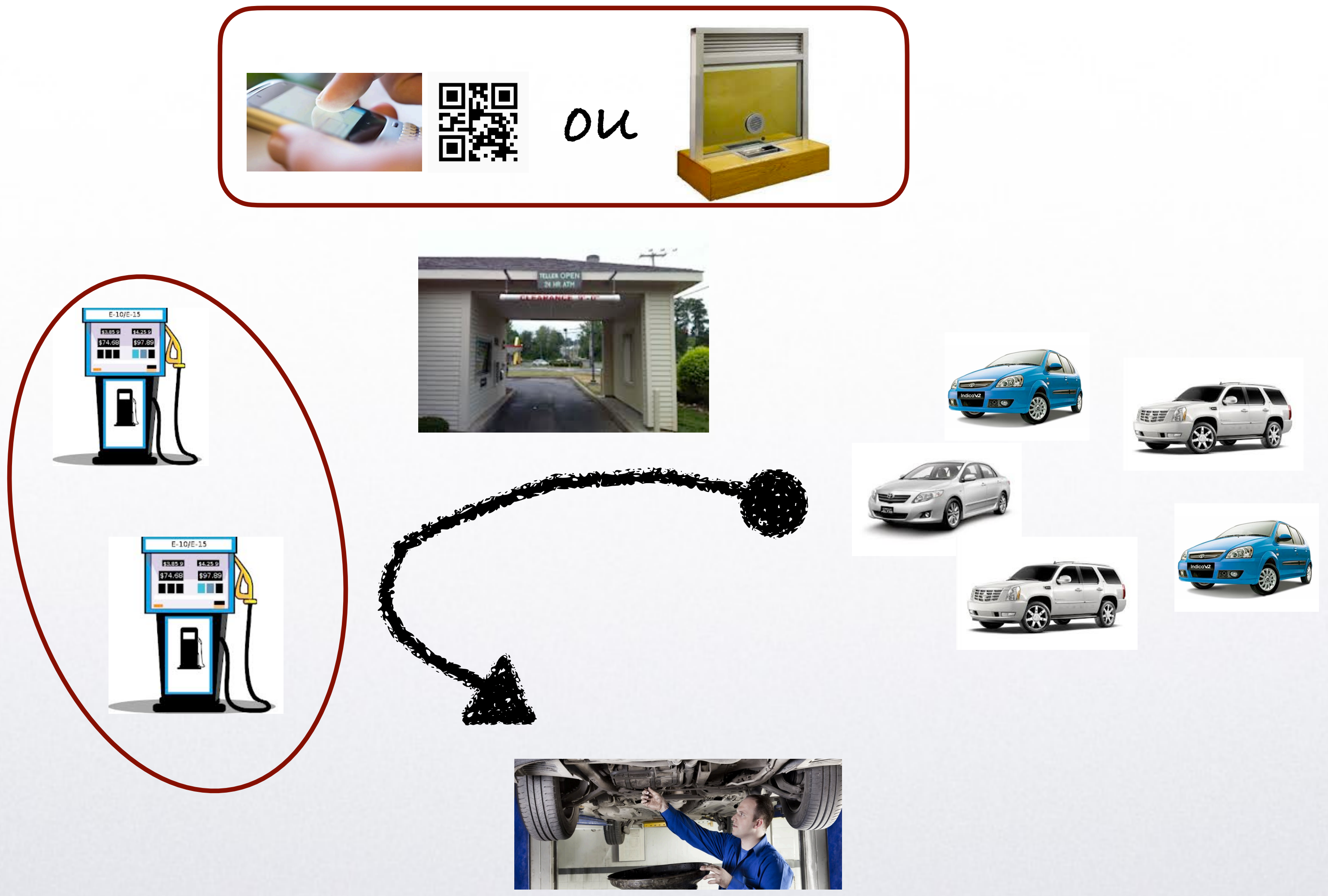
Abstract: The development of Coloured Petri Nets (CP-nets or CPN) has been driven by the desire to develop a modelling language – at the same time theoretically well-founded and versatile enough to be used in practice for systems of the size and complexity found in typical industrial projects. To achieve this, we have combined the strength of Petri nets with the strength of programming languages. Petri nets provide the primitives for describing synchronisation of concurrent processes, while programming languages provide the primitives for definition of data types and manipulation of their data values.

The paper focuses on the practical use of Coloured Petri Nets. It introduces the basic ideas behind the CPN language, and it illustrates how CPN models can be analysed by means of simulation, state spaces and condensed state spaces. The paper

Ao invés de uma lista, vamos nos concentrar em um exercício de modelagem, o do posto de gasolina, mas agora com requisitos adicionais.



Modelar esse sistema em redes CPN (sem um modelo prévio em redes P/T).



Na próxima aula vamos discutir esse processo de modelagem e a síntese de uma rede CPN. Em seguida vamos falar sobre a análise de propriedades tanto nas redes P/T quanto nas redes CPN e HLPN.