PMR 5237

Modelagem e Design de Sistemas Discretos em Redes de Petri

Aula 2: O processo de modelagem

Prof. José Reinaldo Silva
reinaldo@usp.br

Ferramentas de software

PIPE (Windows, Linux, Mac) v. 4.3.0 (2015)

www.softpedia.com/get/Others/Home-Education/PIPE2.shtml

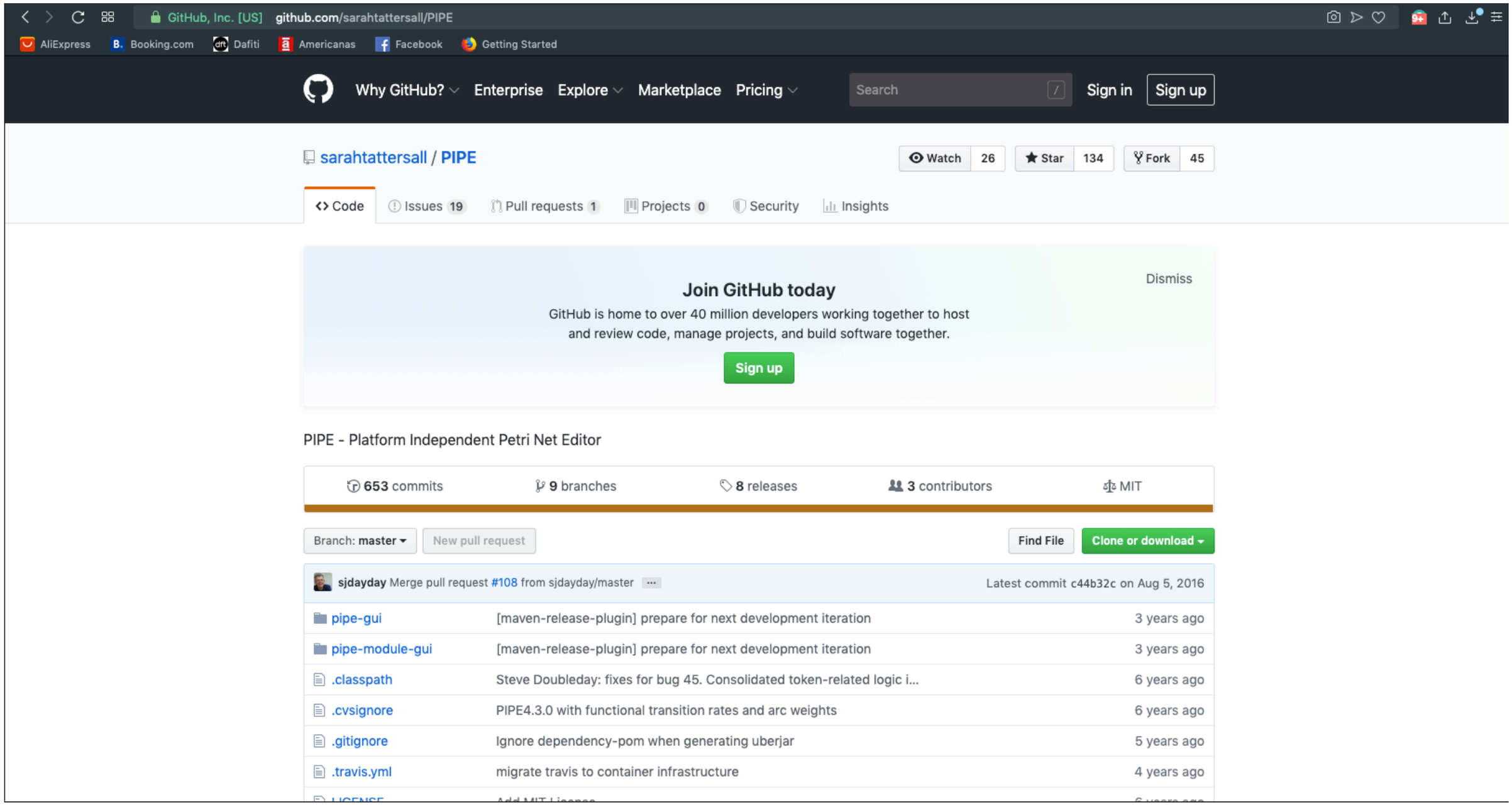
HPSIM (Windows)

CPN Tools

GHENeSys

http://code.google.com/p/ghenesys/

github.com/sarahtattersall/PIPE



www.renew.de

www.renew.de

AliExpress | Booking.com | Dafiti | Americanas | Facebook | Getting Started

Download

Renew is licensed under the GNU Lesser Public License (LGPL), for details see our license terms. By downloading Renew you agree to this terms.

Running Renew requires a Java 1.7 runtime environment (JRE). The installation instructions contain hints where you can find a suitable Java virtual machine.

Base, Source and Plugins

Renew is currently available in version 2.5 (since 2016-06-16).

- Base package (zip, 5.7 MByte) includes core plugins as executable jar files, the documentation and sample nets.
- Source package (zip, 25.5 MByte) includes source files for core and optional plugins of Renew (libraries necessary to build Renew using ant are not included, see the installation instructions).
- Mac OS X application bundle including several optional plugins (7.8 MByte).
- Deb-based package (Debian/Ubuntu/Mint) -- see the installation instructions.
- Optional plugins for the current version.

You can also download old versions of Renew. However those are no longer supported.

Documentation

The main documentation is the user guide. It offers information on installing, upgrading and the release history, introduces the reference nets formalism from a practical perspective with many examples and provides in-depth treatment of the Renew tool.

- User guide in PDF format (1.4 MByte).
- Installation instructions (also included in the user guide and in doc/README)

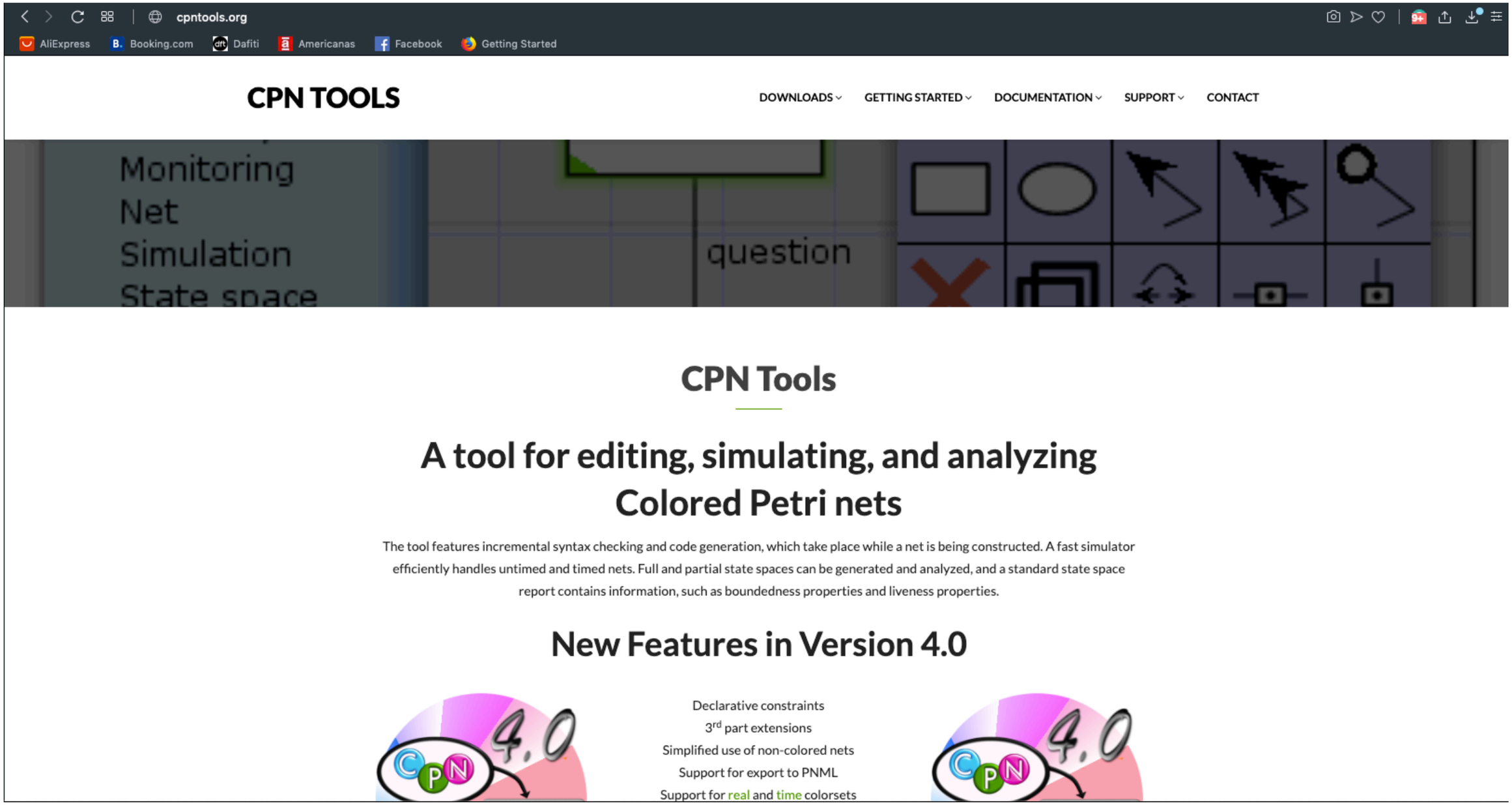
Renew Packages

Renew Analysis

(experimental)
This package includes the Lola Plugin, with which PT-Nets can be analysed.

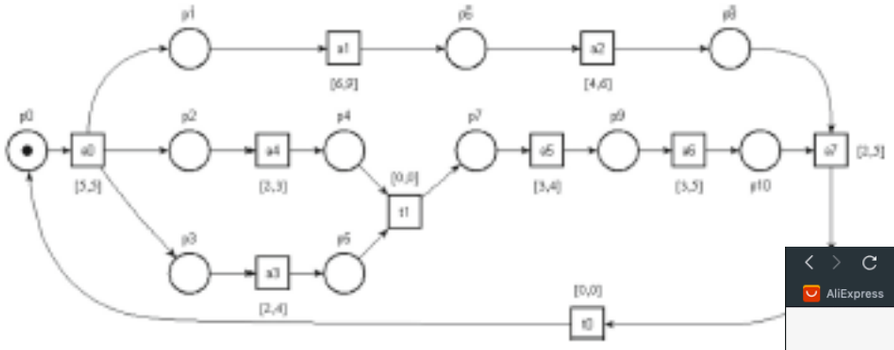
- Linux 64 (zip, 13.4 MByte)
- Linux 32 (zip, 12.9 MByte)
- Windows 64 (zip, 14.7 MByte)
- Windows 32 (zip, 12.9 MByte)
- Mac OS X 64 (zip, 9.3 MByte)

cnptools.org

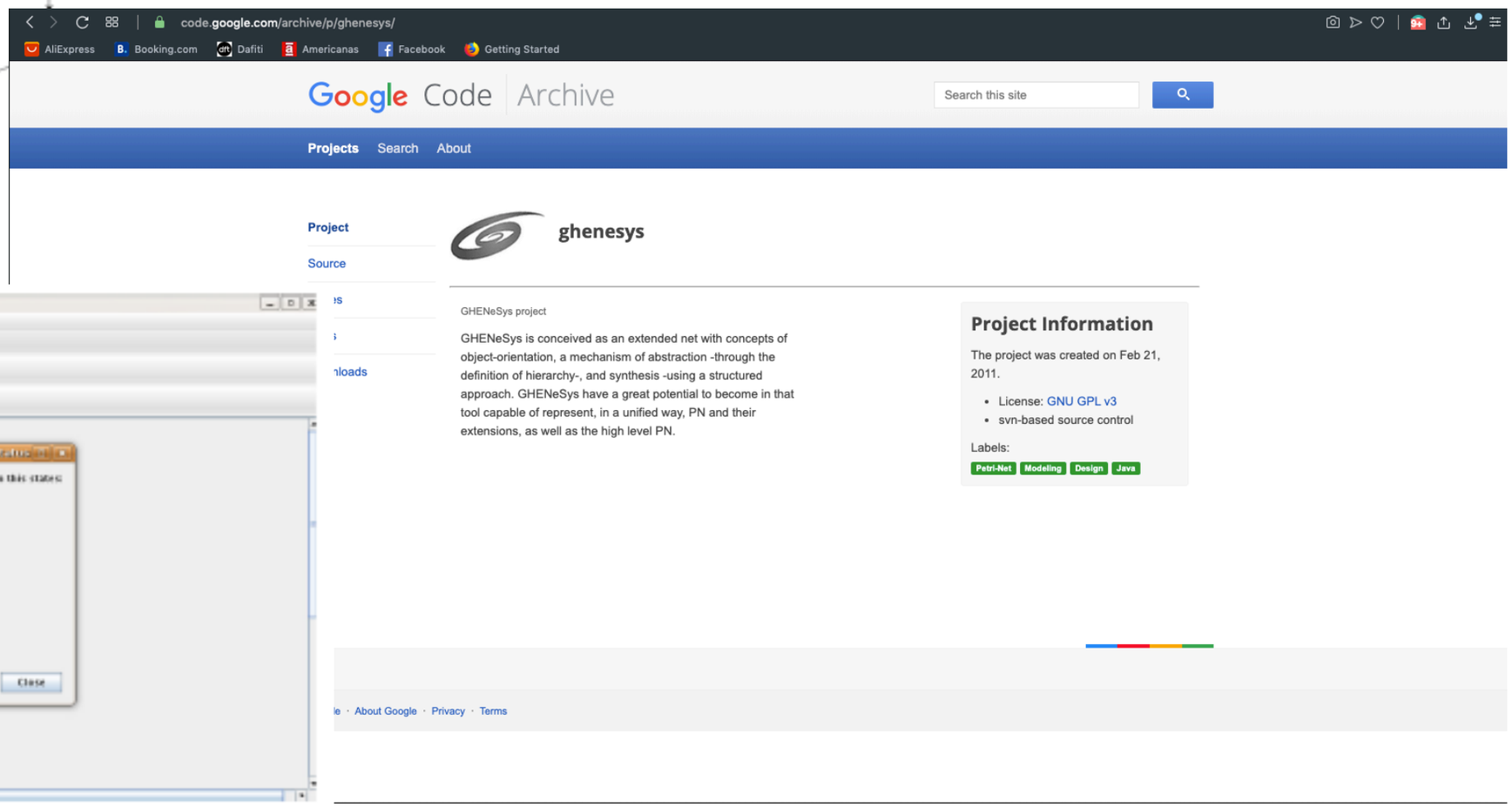


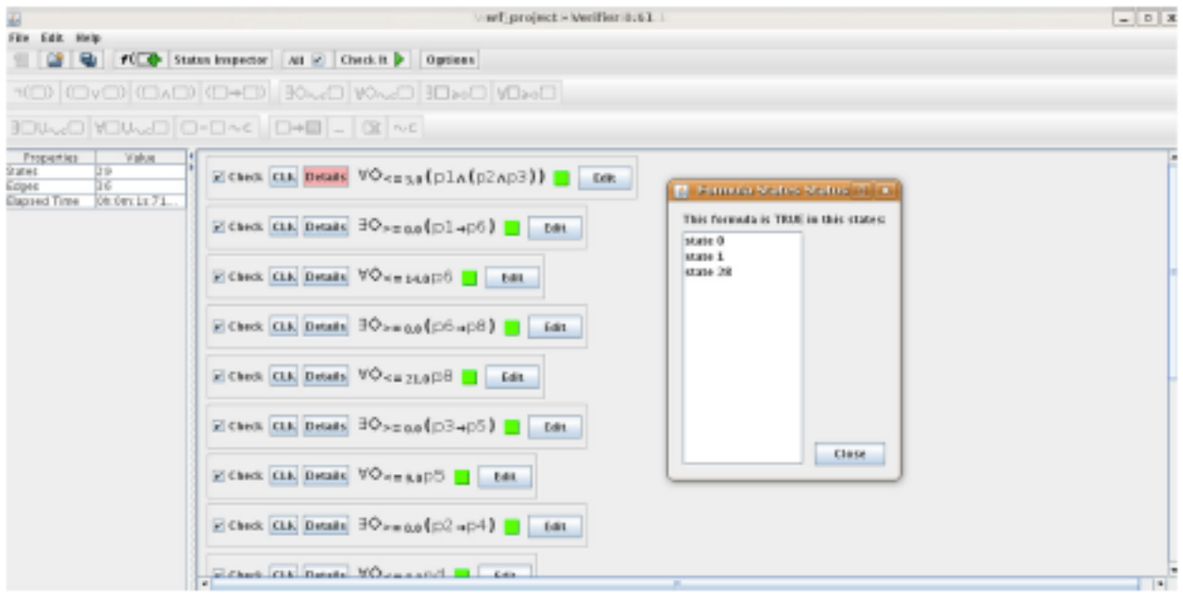
GHENeSYs (**G**eneral **H**ierarchical **E**nhanced **Ne**t **Sys**tem)

Unified Petri net - ISO/IEC 15.909



http://code.google.com/p/ghenesys/

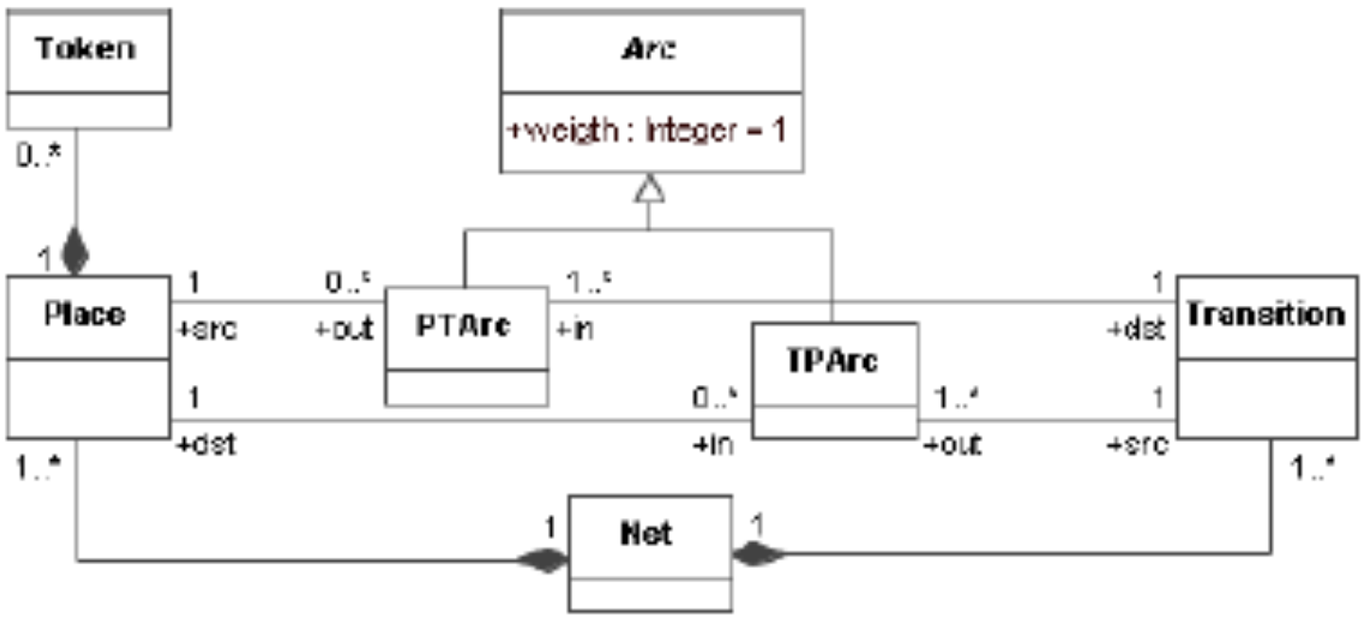




Representação gráfica

Meta-modelo da rede de Petri

μ_5 :



Wachmuth, G.; Metamodel Adaptation and Model Co-adaptation, Atlantic Modeling (AtlanMod), INRIA, Nantes, France, http://www.emn.fr/z-info/atlanmod/index.php/Emfatic#KDM_1.0

A Generic View of Petri Nets

To better understand the basic concepts of Petri nets we will start with a generic approach based on a meta-model, where a PN will be revisited by its main elements and the connection they have with each other. We no longer use a description of the state of each interpreted element. In fact, from now on we will depart from "interpretations" to approach the formal model.

Redes de Petri: Definição

Definition

Definition 1] Uma rede de Petri é um grafo direcionado, simples, bipartido e conexo, representado pela n-upla $N = (S, T; F)$, onde S é um conjunto de estados $\{s_i\}$, T é um conjunto de transições $\{t_j\}$, e F é uma relação de transição (o relação de fluxo), tal que:

i) $S \cap T = \emptyset$ e $S \cup T \neq \emptyset$;

ii) $F \subseteq (S \times T) \cup (T \times S)$;

iii) $dom(F) \cup ran(F) = S \cup T$, onde

$dom(F) = \{x \in (S \cup T) \mid \exists y \in (S \cup T).(x, y) \in F\}$,

$ran(F) = \{y \in (S \cup T) \mid \exists x \in (S \cup T).(x, y) \in F\}$.

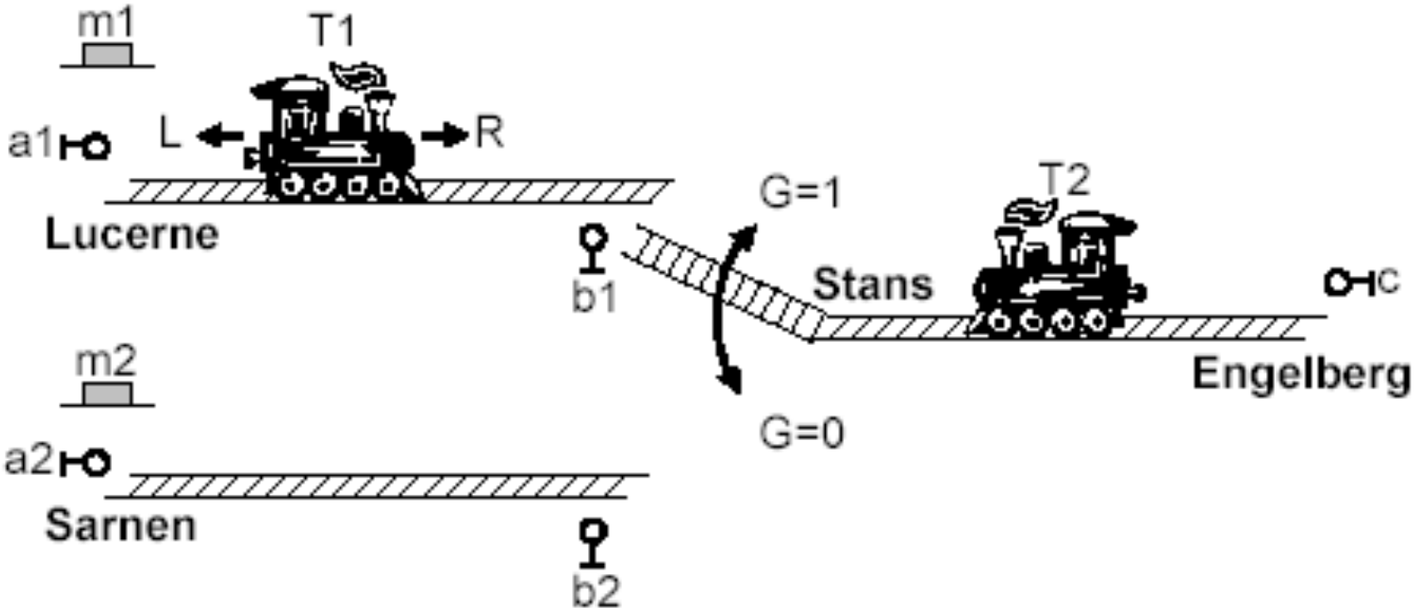
Princípios para modelagem em Redes de Petri

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

Como usar estes princípios na modelagem de um problema real (que pode ser grande o suficiente para não permitir a listagem literal de cada estado atômico.

Exemplo: manobrando linhas de trem



A especificação oficial do problema

1 The Winter Train Problem

We consider two trains T1 and T2 transporting skiers from Sarnen and Lucerne to Engelberg. Because there is only one ground rail track from Stans to Engelberg, at most one train might be between these two villages at any time. There is a switch in Stans, which either connects the track between Sarnen and Engelberg xor the track between Lucerne and Engelberg. After the train conductor has pressed a button m in (Sarnen *or* Lucerne), its train moves to Engelberg, but might have to wait in Stans until the other train has left the critical section. Once arrived in Engelberg, the train waits for 100s and then returns. The sensors a1, a2, b1, b2 and c indicate the presence of a train with the value 1, otherwise, the value is 0. The switch in Stans is accessed through a variable G, as indicated in the picture.
Finally, the motion of the trains is regulated by assigning 'R', 'L' or 'S' to the train, to move right, left, or stop, respectively.

O processo de modelagem em RdP

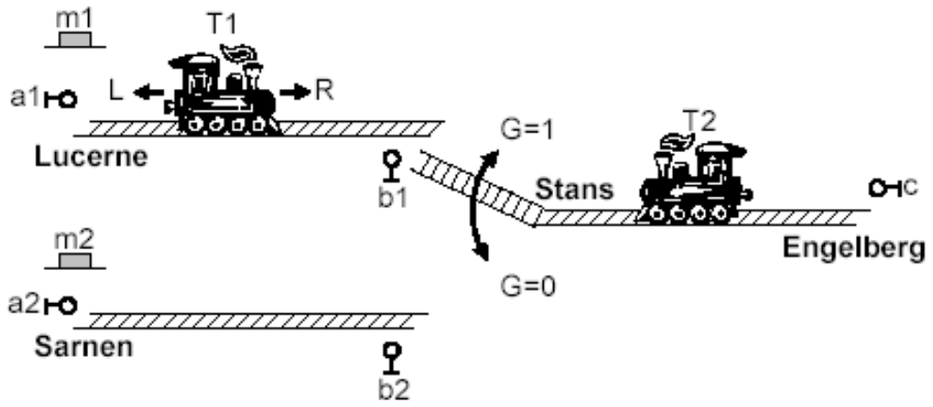
Onde podem estar os trens e como se reconhece estes estados?

i) Defina o estado inicial claramente;

ii) Identifique os estados de cada trem separadamente;

iii) Note que uma combinação livre destes estados não é possível devido às restrições dos trilhos;

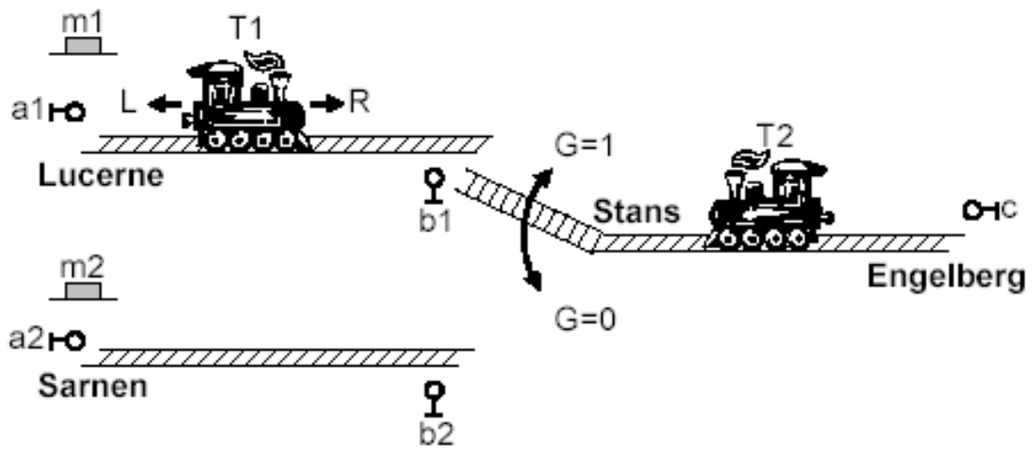
iv) Veja quais as combinações possíveis que utilizam os sensores



Passo 1: definindo o estado inicial

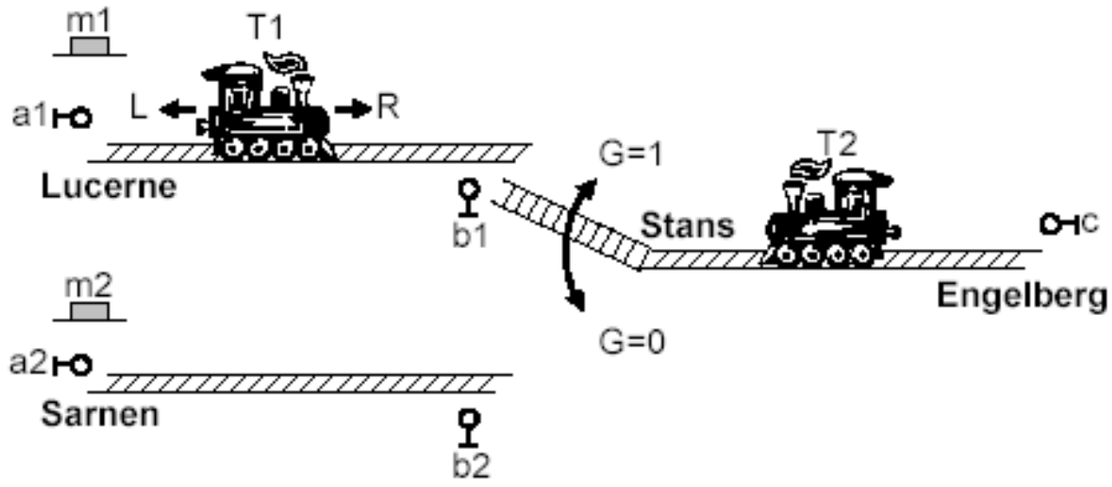
O estado inicial pode ser definido arbitrariamente desde que não viole as condições iniciais do problema. Para este caso identifique bons candidatos ao estado inicial. Os demais estados decorrem desta situação ou são "gerados" por este.

Convencionalmente vamos admitir que o estado inicial é dado pelo trem T1 em Lucerne, prestes a sair em direção a Engelberg e o trem T2 em Engelberg, prestes a sair para Sarnen, e pelo sinal m1 do operador ordenando a saída de T1.



Passo 2: identificando os estados

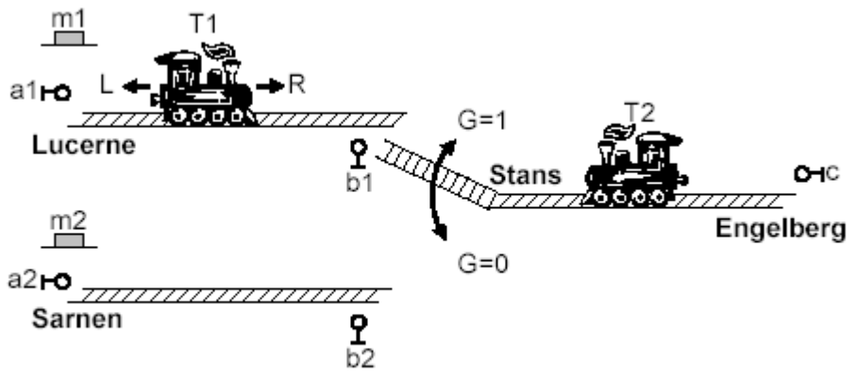
Os estados do sistema são determinais pela Posição dos dois tens. Portanto, parece uma boa idéia analisar cada trem em separados depois ver os estados proibidos, isto é, aqueles estados indesejados, onde os trens estão ambos no trecho unificado Stans-Engelberg.



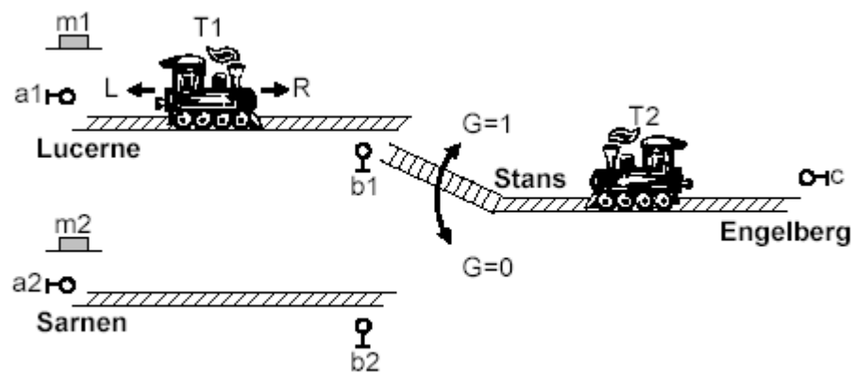
Identificando os estados do trem T I

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;



Identificando os estados do trem T2



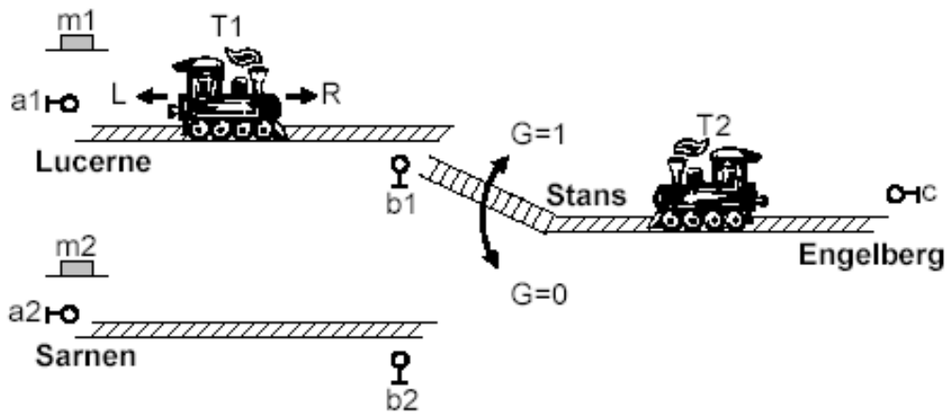
Movimento do trem T2

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;

Identificando as transições

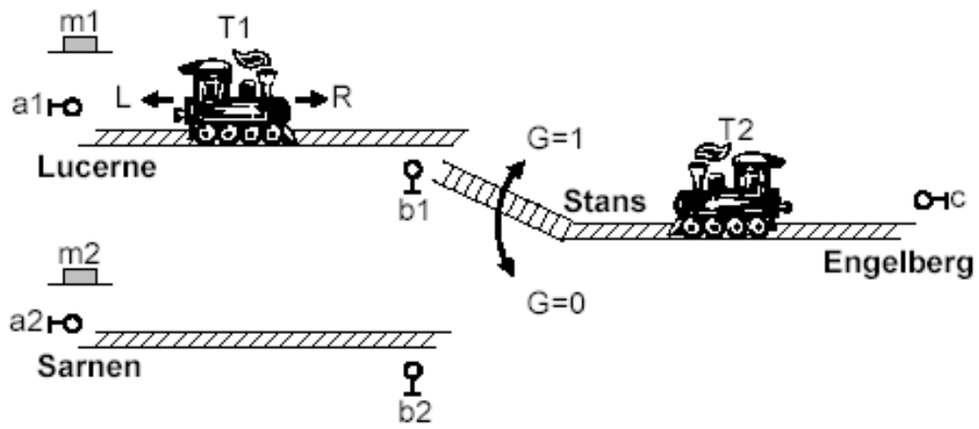
Cada trem sai de uma cidade e trafega no trecho livre sozinho, até a cidade de Stans onde fica o chaveamento. Como trafegam em sentido contrário (sempre) há um movimento preferencial para que um deles libere o trecho compartilhado, depois o chaveamento é modificado e o outro entra também no trecho desobstruído pelo trem anterior.

Quantos eventos são necessários para representar univocamente o problema?.



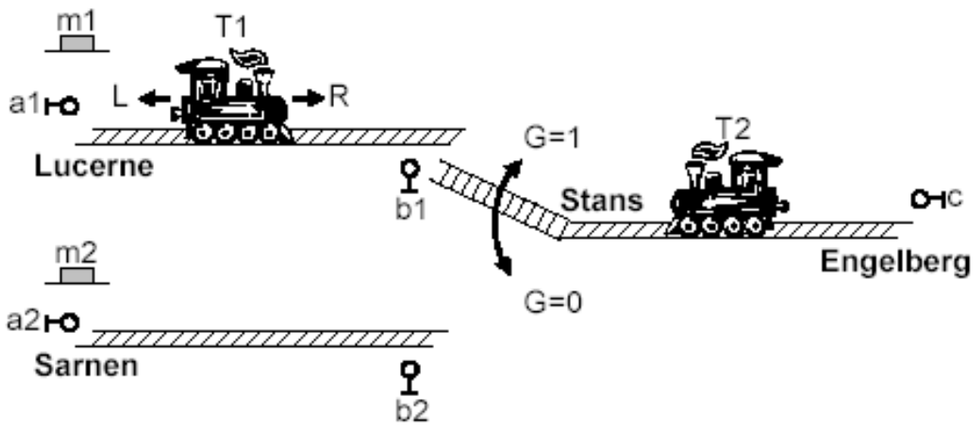
Transições de T1

T0 – Trem T1 sai do ponto Lucerne;
T1 – Trem T1 chega em Stans vindo de Lucerne;
T2 – Trem T1 entra no trecho unificado;
T3 – Trem T1 chega em Engelberg;
T4 – Trem T1 sai de Engelberg para Stans;
T5 – Trem T1 chega em Stans vindo de Engelberg;
T6 – Trem T1 entra no trecho Stans-Lucerne;
T7 – Trem T1 chega em Lucerne;

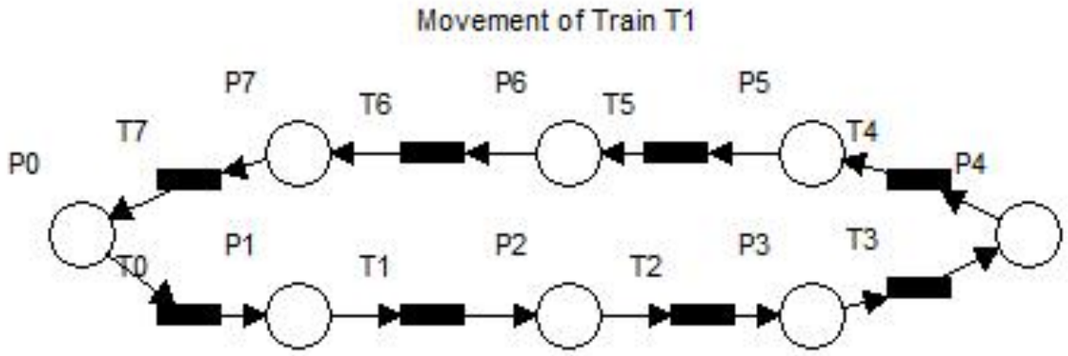


Transições de T2

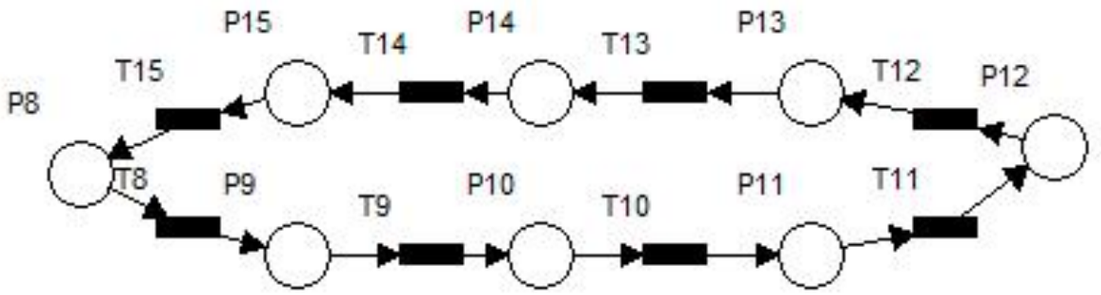
T8 – Trem T2 sai do ponto de Engelberg;
T9 – Trem T2 chega em Stans vindo de Engelberg;
T10 – Trem T2 entra no trecho Stans-Sarnen;
T11 – Trem T2 chega em Sarnen;
T12 – Trem T2 sai de Sarnen para Stans;
T13 – Trem T2 chega em Stans vindo de Sarnen;
T14 – Trem T2 entra no trecho unificado Stans-Engelberg;
T15 – Trem T2 chega em Engelberg;

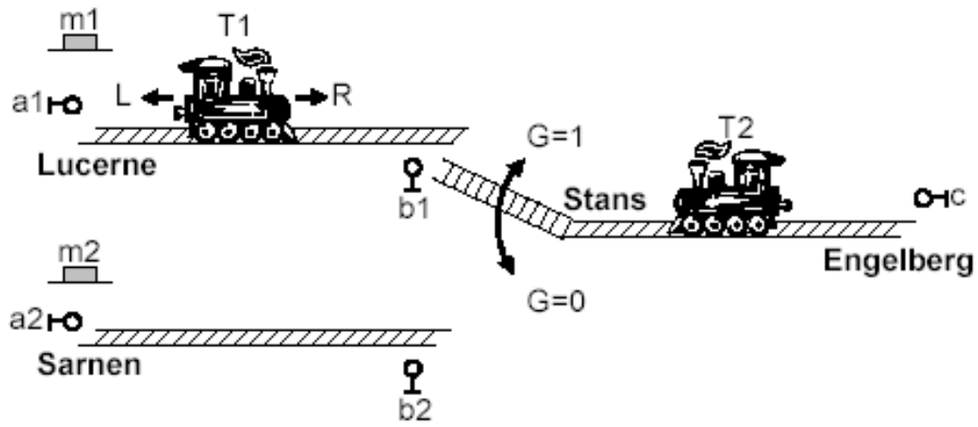


O problema de automação e controle



Nos diagramas ao lado temos o modelo gráfico do movimento de cada trem (um esquema cuja interpretação do significado de lugares e transições se encontra nas transparências anteriores). O problema de automação aqui é do tipo semáforo, no sentido que somente um dos trens pode estar no trecho unificado de cada vez, e de sincronismo, dado que, se um dos trens (T1) faz o trajeto de Lucerne a Engelberg, ao voltar deve encontrar o gate G na posição 1. Similarmente o outro trem (T2) deve encontrar este mesmo gate na posição G=0.





Modelando o problema de automação

Supondo que os trens fazem repetidamente o percurso entre estas cidades, o sistema global é cíclico, isto é, retorna ao estado inicial, e repete sempre a mesma seqüência de ações. Trata-se de um sistema que satisfaz as condições ideais para um processo de automação.

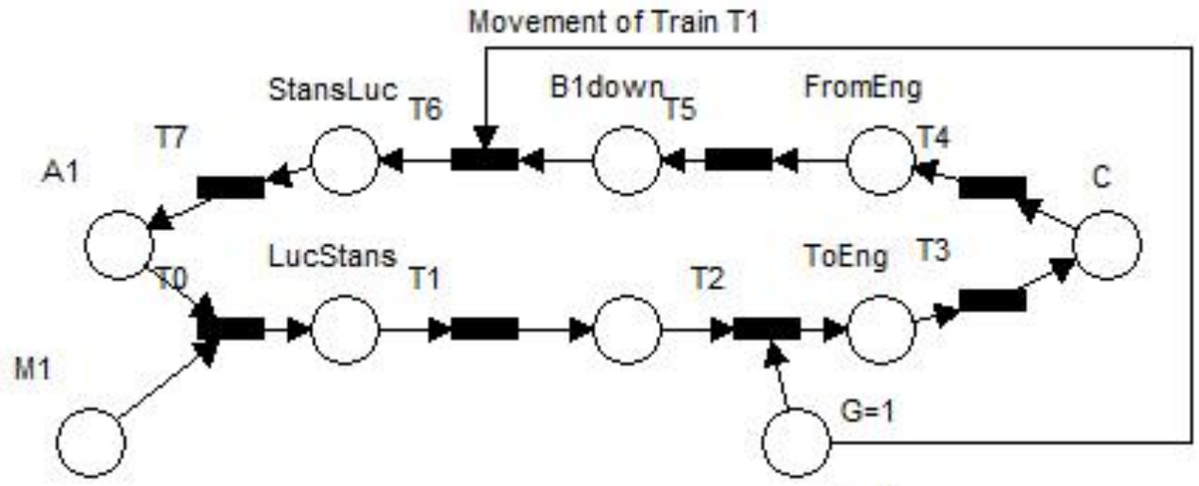
Como vamos mostrar isto?

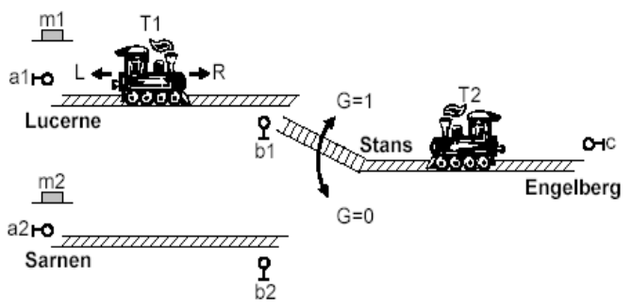
Lista de exercícios (Exerc. 1)

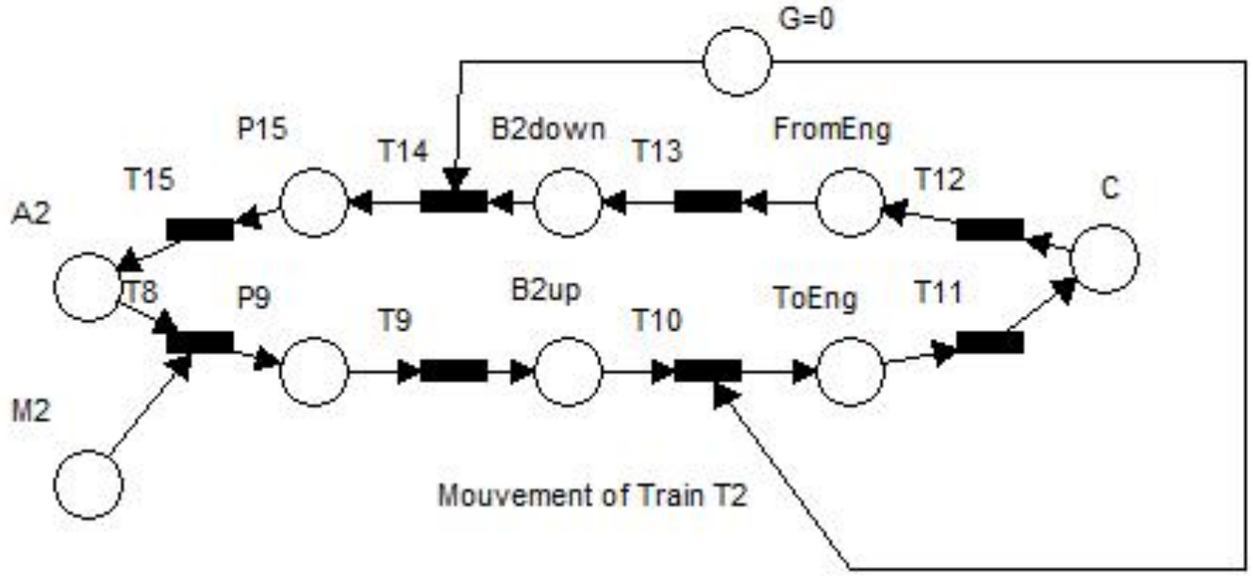
Represente graficamente este problema no PIPE ou no GHENeSys e denote o estado inicial com marcas nos lugares correspondentes. Faça o sistema disparar os estados independentes um número grande de vezes (comparado ao número de eventos do sistema) e mostre as características acima.

Usando o gate para sincronizar o movimento

Modelamos então o estado do gate G e sua influência no movimento de cada trem

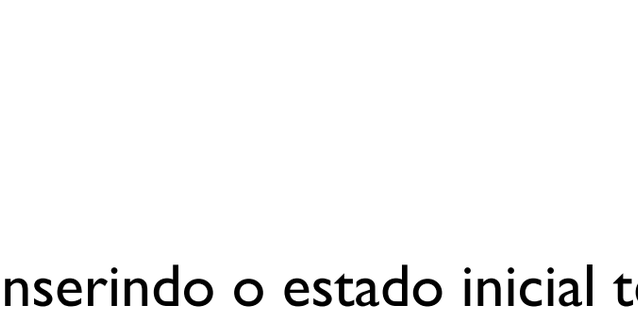


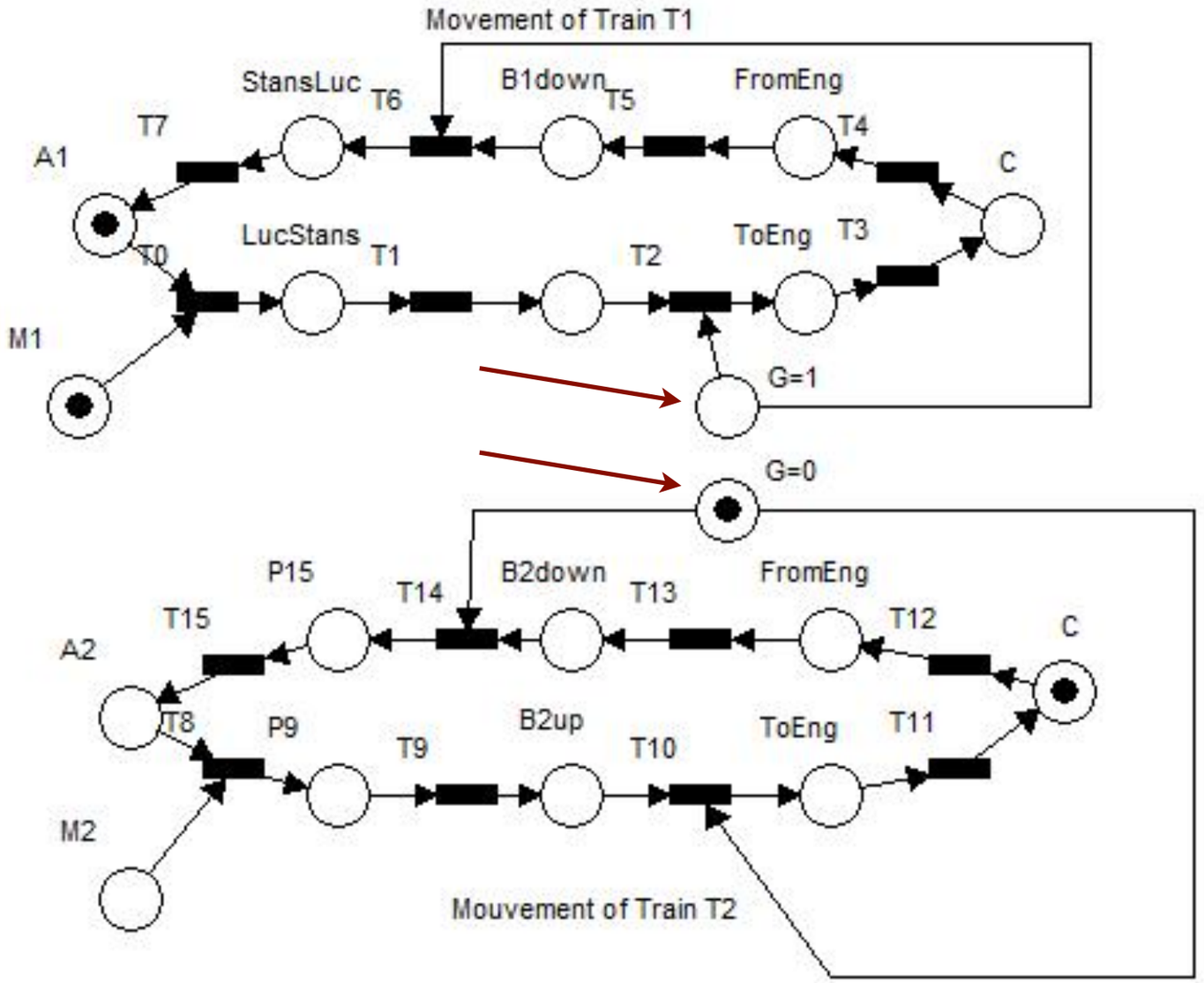




Síntese do modelo obtido

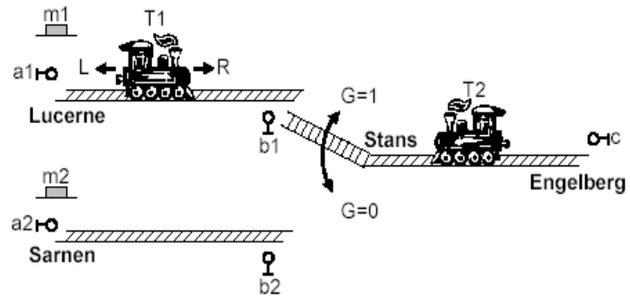
Inserindo o estado inicial temos o problema parcialmente modelado, isto é, apenas com a sincronização resolvida. Mas note que os lugares apontados pelas setas representam estados do mesmo gate G. Portanto se um deles é marcado automaticamente desmarca o outro, configurando um conflito





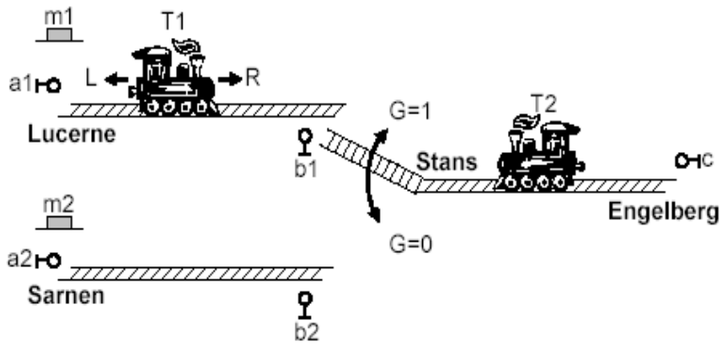
O chaveamento (mutex)

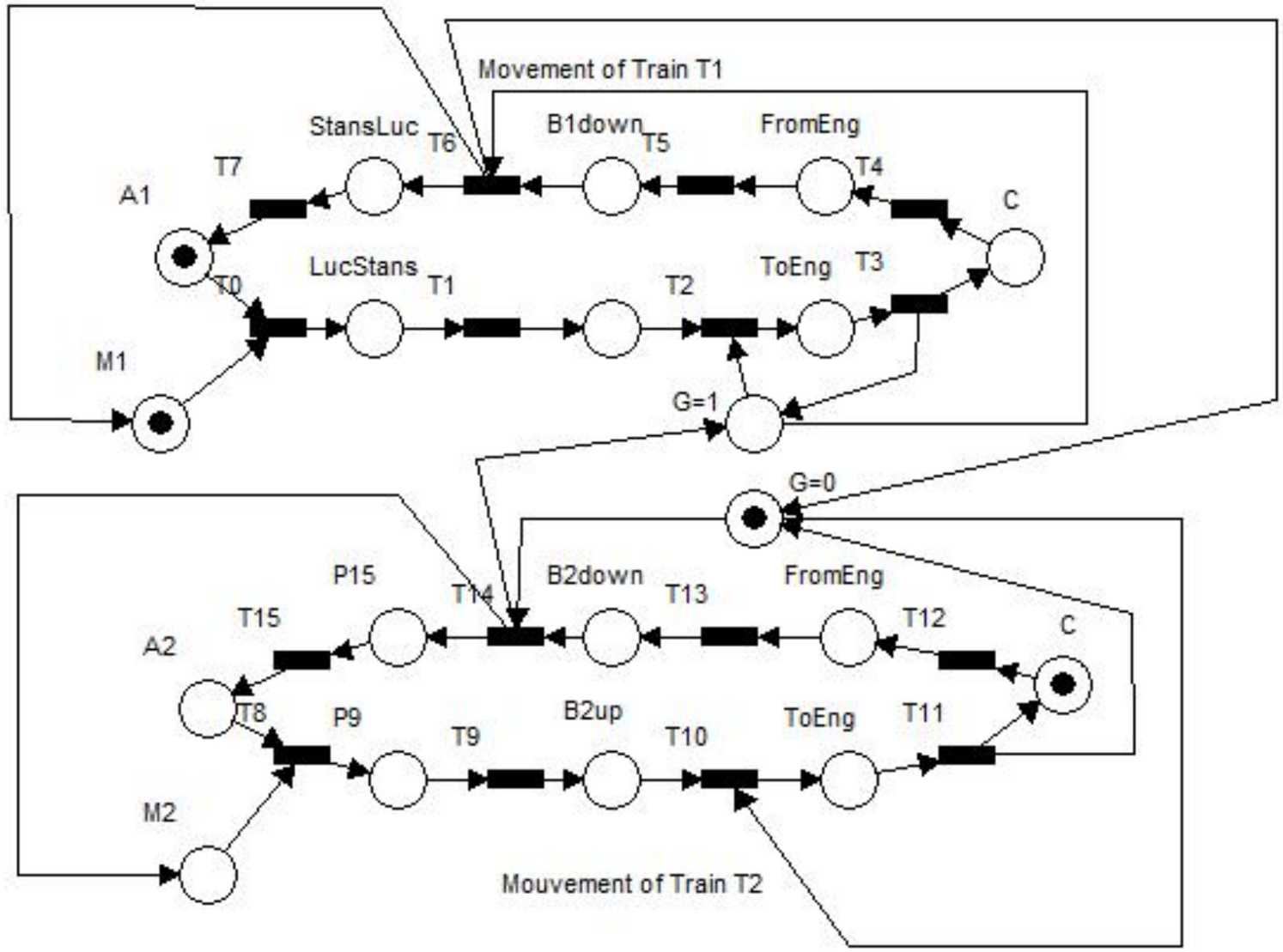
Até este ponto modelamos completamente o movimento dos dois trens separadamente. O problema agora é modelar o controle, que é responsável pelo estado do gate G. Neste caso o gate estará sempre preparado para o trem que está no trecho unificado (situação de maior risco). Os sensores b1 e b2 existem exatamente para provocar a parada dos trens T1 e T2 respectivamente e esperar pelo chaveamento de G. Este fica nesta posição até que o trem que está no trecho unificado saia e chaveia para o outro trem. É portanto uma situação de conflito que deve ser resolvida pelo controle do gate G com auxilio dos sensores.



O modelo completo

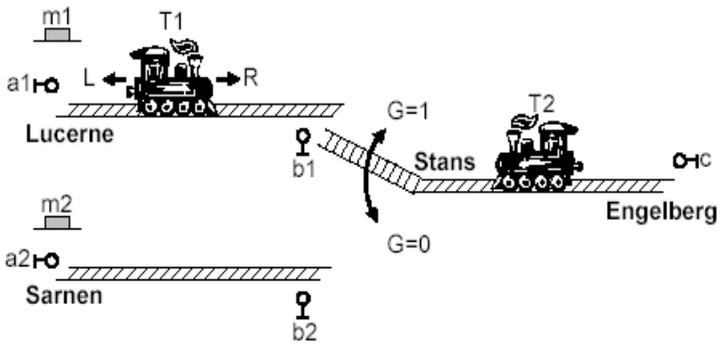
Garantindo a alternância de marcação do mutex, e também que o modelo seja cíclico, isto é, que retorna ao estado inicial depois de alguns disparos, temos o modelo completo.

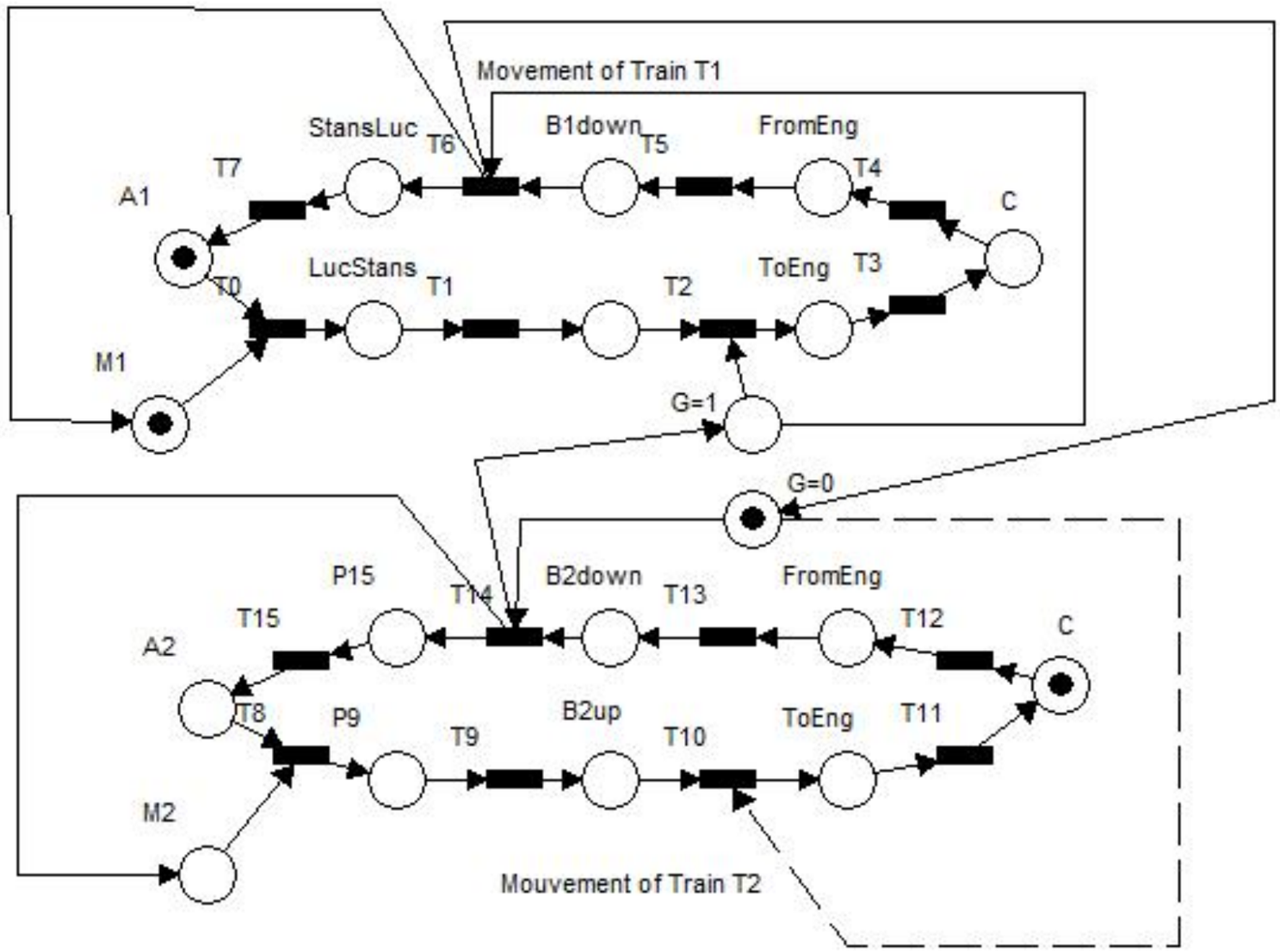




O uso de extensões

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).





Lista de exercícios: Exec. 2

Suponha que depois de fazer a modelagem e a simulação (jogo de marcas) em Redes de Petri, você deva fazer uma apresentação dos resultados para o chefe da companhia e para os supervisores garantindo que o sistema automatizado funcionará "sem erros". Qual seria o argumento básico (mas intuitivo)?

Lista de exercícios (Exec. 2)

Modele o problema completo no PIPE, primeiro com todos os arcos normais e depois com os gates. Mostre que as redes são equivalentes, isto é, geram os mesmos estados na mesma sequência. Compare ainda as matrizes de incidência nos dois casos.

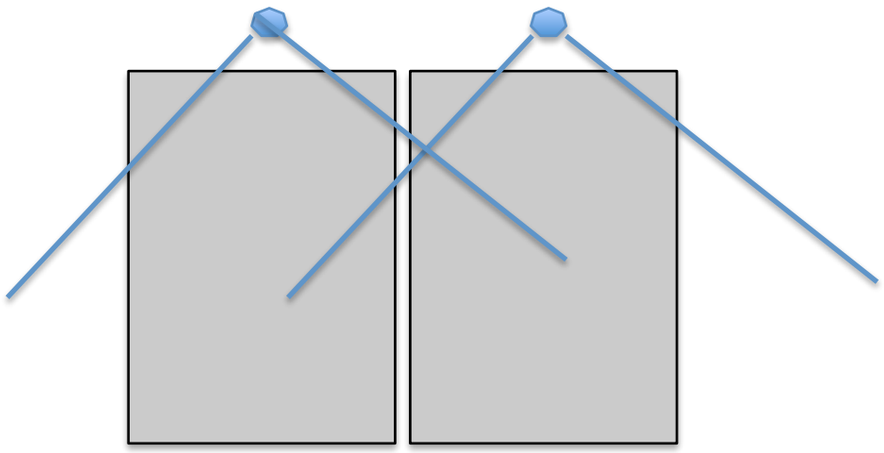
Exec. 3

Identifique na matriz de incidência, nos dois casos, o conflito e a forma como este é "resolvido", isto é, o sistema não tem nenhum estado cujo sucessor não esteja plenamente definido (pelo estado do gate G). Explique o funcionamento do controle do trem. Note que o controle da linha compartilhada poderia ser totalmente automatizado.

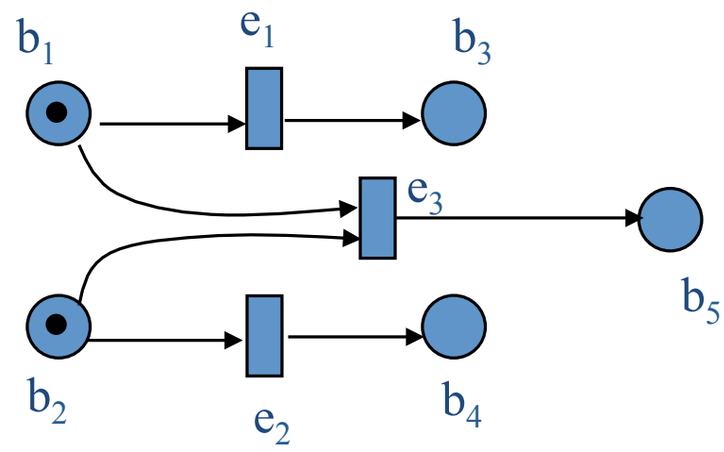
Princípio da Representação Algébrica

Todo sistema modelado em Redes de Petri admite uma e somente representação algébrica, que é dada pelas matrizes que representam a estrutura do modelo (estados e transições distribuídas) e por uma equação de estado que representa a evolução dos estados pela ocorrência das transições habilitadas.

Usando a representação matricial



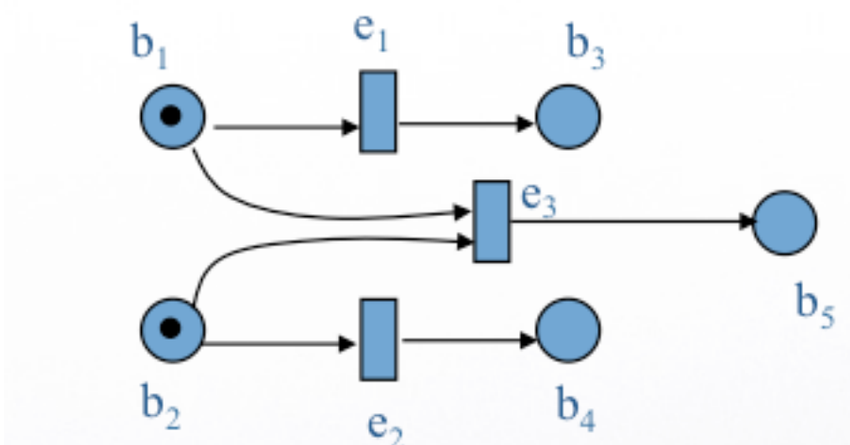
No caso das redes se faz uma tabela de dupla entrada das transições contra os lugares. Cada elemento da matriz indica se o lugar é incidente, -1, emergente, 1, ou se não há conexão de nenhum tipo, 0.



$$\begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \end{array} \begin{array}{c} \begin{array}{ccccc} b_1 & b_2 & b_3 & b_4 & b_5 \end{array} \\ \begin{bmatrix} -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ -1 & -1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

A notação matricial pode agora ser estendida à marcação. Neste caso um estado de marcas M será denotado por um vetor de dimensão igual ao número total de marcas da rede. Cada elemento deste vetor terá como valor um inteiro positivo denotando o número total de marcas no respectivo lugar

$$M_i = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \begin{matrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \end{matrix}$$



	b_1	b_2	b_3	b_4	b_5
e_1	−1	0	1	0	0
e_2	0	−1	0	1	0
e_3	−1	−1	0	0	1

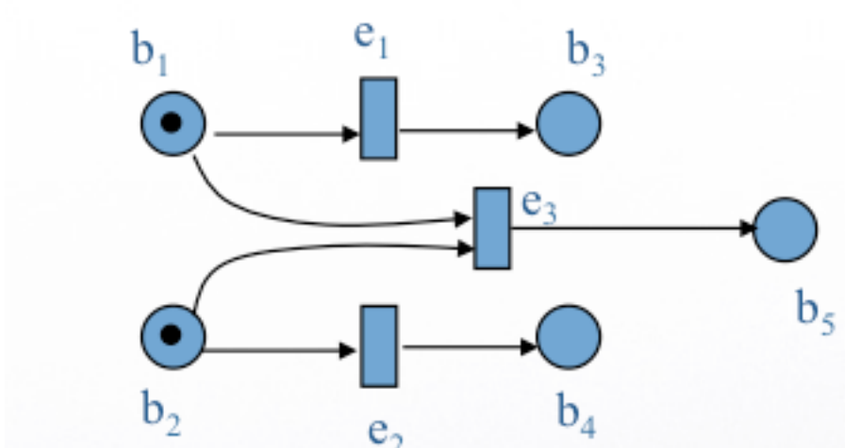
Evolução Estado/Transição

Def. 7

Seja uma rede de Petri $N = (S, T; F)$. Se uma transição elementar $t \in T$ habilitada em um estado $M \subseteq S$ ocorre, o estado M evolui para o estado $M' = (M \setminus {}^\bullet t) \cup t^\bullet$.

A evolução das marcas foi tratada na aula passada usando a notação de conjuntos, mostrada pela definição 7 acima. Cabe agora transferir este resultado para a notação matricial.

Em primeiro lugar note que a matriz de incidência do exemplo ao lado pode ser transformada em uma soma onde uma parcela representa o fluxo incidente e a outra o fluxo emergente. Assim temos que



$$A = A^- + A^+$$ onde

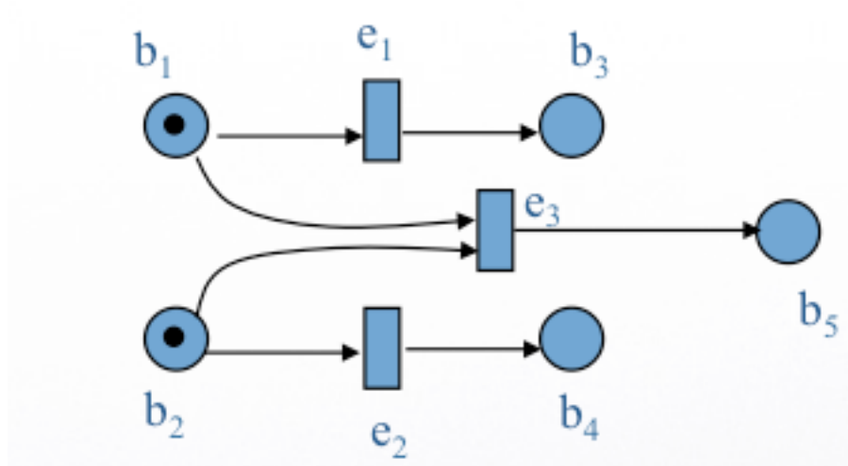
.

$$A^- = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 \end{pmatrix} \text{ e } A^+ = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

.

	b_1	b_2	b_3	b_4	b_5
e_1	−1	0	1	0	0
e_2	0	−1	0	1	0
e_3	−1	−1	0	0	1

Similarmente um passo *T* pode ser representado por um vetor unimodular com a dimensão do número de transições existentes na rede, onde cada elemento o vetor term valor unitário se a respectiva transição está habilitada no estado corrente e zero em caso contrário. Este vetor também é chamado vetor de habilitação. Para o exemplo ao lado temos que,



$$T = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{matrix} e_1 \\ e_2 \\ e_3 \end{matrix}$$

	b_1	b_2	b_3	b_4	b_5
e_1	−1	0	1	0	0
e_2	0	−1	0	1	0
e_3	−1	−1	0	0	1

A equação de estado

Note ainda que para um dado passo genérico *T*,

$$\bullet T = (A^-)^T T \text{ e } T\bullet = (A^+)^T T$$

Def. 7

Seja uma rede de Petri $N = (S, T; F)$. Se uma transição elementar $t \in T$ habilitada em um estado $M \subseteq S$ ocorre, o estado M evului para o estado $M' = (M \setminus \bullet t) \cup t\bullet$.

Portanto, se comparamos com a Def.7 da aula passada temos que,

$$M' = M + \bullet T + T\bullet = M + (A^-)^T T + (A^+)^T T = M + A^T T$$

A equação de estado

Finalmente, podemos ter a equação que dá o fluxo de marcas (equação de estado) expressa na forma matricial como,

$$M_i = M_0 + A^T \sum_0^{i-1} T_j = M_0 + A^T \sigma_{i-1}$$

Lista de exercícios: Exec. 4

Mostre que se o vetor de habilitação usado na equação de estado denotar uma situação de conflito o estado final é inconsistente, isto é, pode ter marcação negativa.

Forward case class

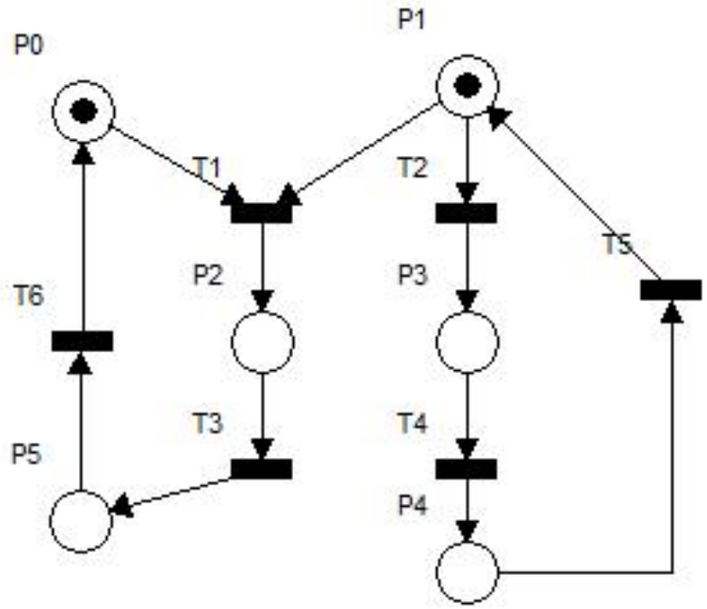
Portanto é possível gerar estados a partir de um estado dado, que pode ser, por exemplo o estado inicial. O conjunto de estados gerados a partir deste gerador é chamado de forward case class e é denotado por $\lceil M_0 \rangle$

Lista de exercícios: Exec. 5

Faça um algoritmo que age sobre a matriz de incidência de uma rede genérica identificando as situações de conflito. Nestes casos o usuário do "jogador de marcas" deve escolher um passo ou vetor de habilitação onde o conflito não esteja inserido.

Deadlocks

A situação de deadlock é caracterizada quando a rede se encontra em um estado que não tem sucessor, isto é, onde nenhuma transição pode ser habilitada. Isto pode ser o resultado de uma má escolha do estado inicial ou da sequencia de estados (das transições disparadas).



Sistema Elementar

Portanto, para efeito de modelagem e análise de sistemas a escolha do estado inicial é sempre muito importante. Definiremos a seguir um tipo de redes de Petri, inserido na classe do que é chamado de redes clássicas.

Definition (8)

Uma rede de Petri elementar é uma n-upla $N = (S, T; F, M_0)$, onde $(S, T; F)$ é uma estrutura de rede como definido anteriormente.

O conjunto de estados que este sistema admite é determinado pela escolha do gerador M_0 e é denotado por $\mathcal{C}_{\mathcal{N}} = \lceil \mathrm{M}_0 \rangle$, que é o seu "forward case class".

Def 9] Seja N=(S,T;F, c_0) um sistema elementar. O case set de N, denotado por $\mathcal{C}_N$, é o conjunto minimal de $\wp$(S) satisfazendo as seguintes condições :

i) $c_0 \in \mathcal{C}_N$;

***ii*) se $c_1 \in \mathcal{C}_N$ e $\exists\, v \subseteq T \mid c_1 |v\rangle$, então $c_1 |v\rangle c_2$, e $c_2 \in \mathcal{C}_N$**

Def. 10] Seja N=(S,T;F, c_0) um sistema elementar. O conjunto de todos os passos deste sistema, denotado por P_N é dado por,

$$P_N = \{v \subseteq T \mid \exists\, c_1 , c_2 \in \mathcal{C}_N \,.\, c_1 [v\rangle\, c_2\}$$

Uma rede elementar N=(B,E;F, c_0) é definida sobre um conjunto de cases $\mathcal{C} \subseteq \mathcal{P}(B)$, chamado *case class* de N.

Def. 11] Definimos a relação de atingibilidade R = (r $\cup$ r^{-1})*, onde
r $\subseteq \mathcal{P}(B) \times \mathcal{P}(B)$ é tal que c **r** $c' \Leftrightarrow \exists \wp \subseteq E$ tal que $c \mid \wp \rangle c'$.

Proposição 1] C é classe de equivalência da relação R.

Demonstração : lista de exercícios. Exec. 6

Proposição 1: $\mathcal{C}$ é uma classe de equivalência de R

Dem]

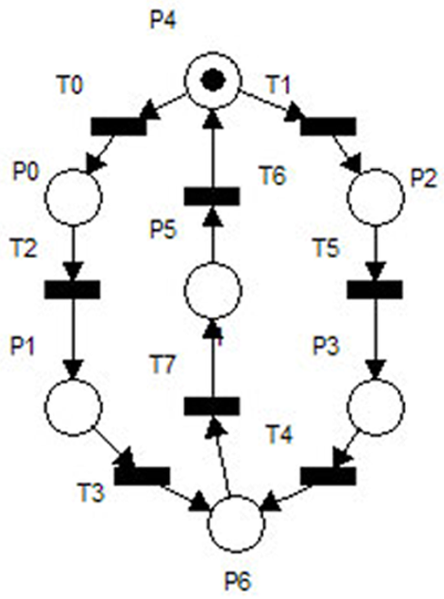
***i*) R é reflexiva. Trivialmente ∀ c, c r c e portanto (c, c) ∈ R.**

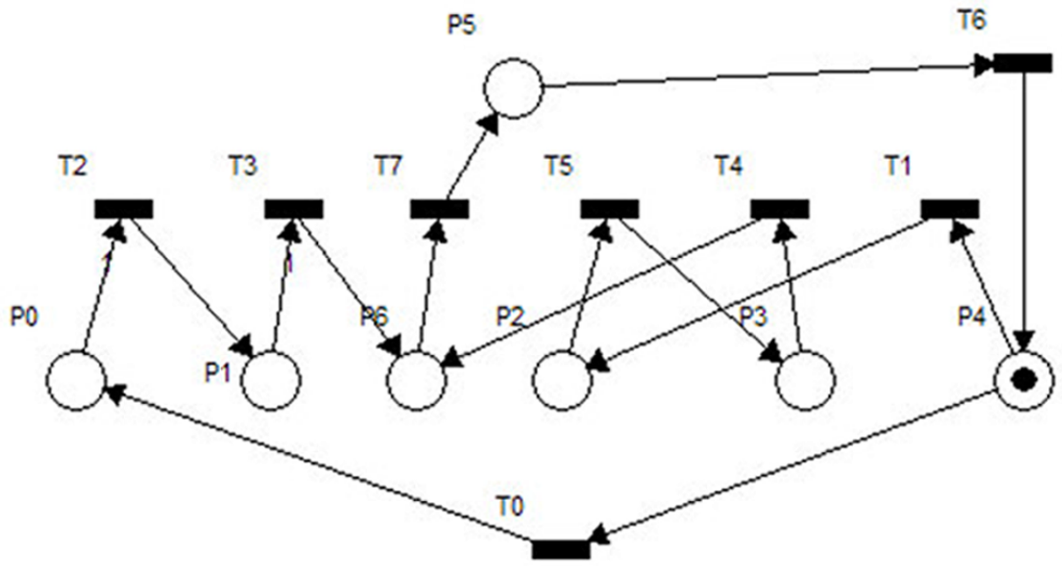
***ii*) R é simétrica. Trivialmente ∀ (c, c') ∈ r, ∃ $\wp^{-1}$ | c' |$\wp^{-1}\rangle$ c e portanto (c', c) ∈ r^{-1} e portanto (c', c) ∈ R.**

***iii*) R é transitiva ...** Dem: Lista de exercícios, Exerc. 6

Representação Gráfica

Cuidado!





Aplicações das Redes de Petri

Seqüenciamento de tarefas (planning)

- trajetórias de AGV´s
- montagem
- problemas modelo (mundo de blocos)
- planejamento reativo

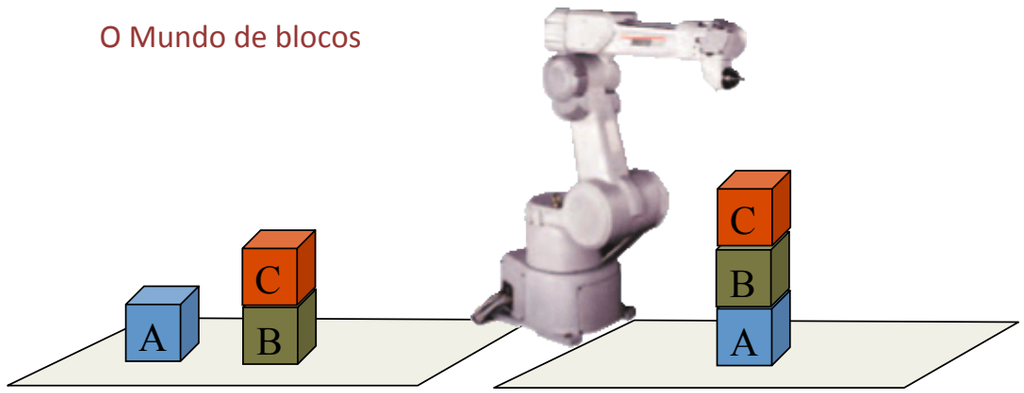
Aplicações em IA Planning e Inteligência de Máquinas

A chamada "inteligência das máquinas" na verdade é reduzida a uma sequencia automatizada de ações (ou movimentos) sem a intervenção de operadores humanos seja de forma direta ou indireta.
Entretanto este mesmo comportamento foi previsto e programado para ser executado quando uma condição ou localidade acontece.
Esta programação é chamada de "planning" e em geral métodos de Inteligência Artificial são usados para permitir que a escolha certa da sequencia de ações seja feita. Redes de Petri podem ser usadas no processo de projeto destes "planos".

IA Planning: o STRIPS

O sistema STRIPS é a estratégia de resolução de problemas mais usada em planning. Note-se que é uma estratégia baseada no método estado-transição e por isso é passível de ser analisada em Redes de Petri. O problema modelo mais conhecido resolvido com o sistema STRIPS é o problema do mundo de blocos.

O Mundo de blocos



Stanford Research Institute Problem Solver

Definition

A STRIPS instance is composed of:

- An initial state;
- The specification of the goal states – situations which the planner is trying to reach;
- A set of actions. For each action, the following are included:
  - preconditions (what must be established before the action is performed);
  - postconditions (what is established after the action is performed).

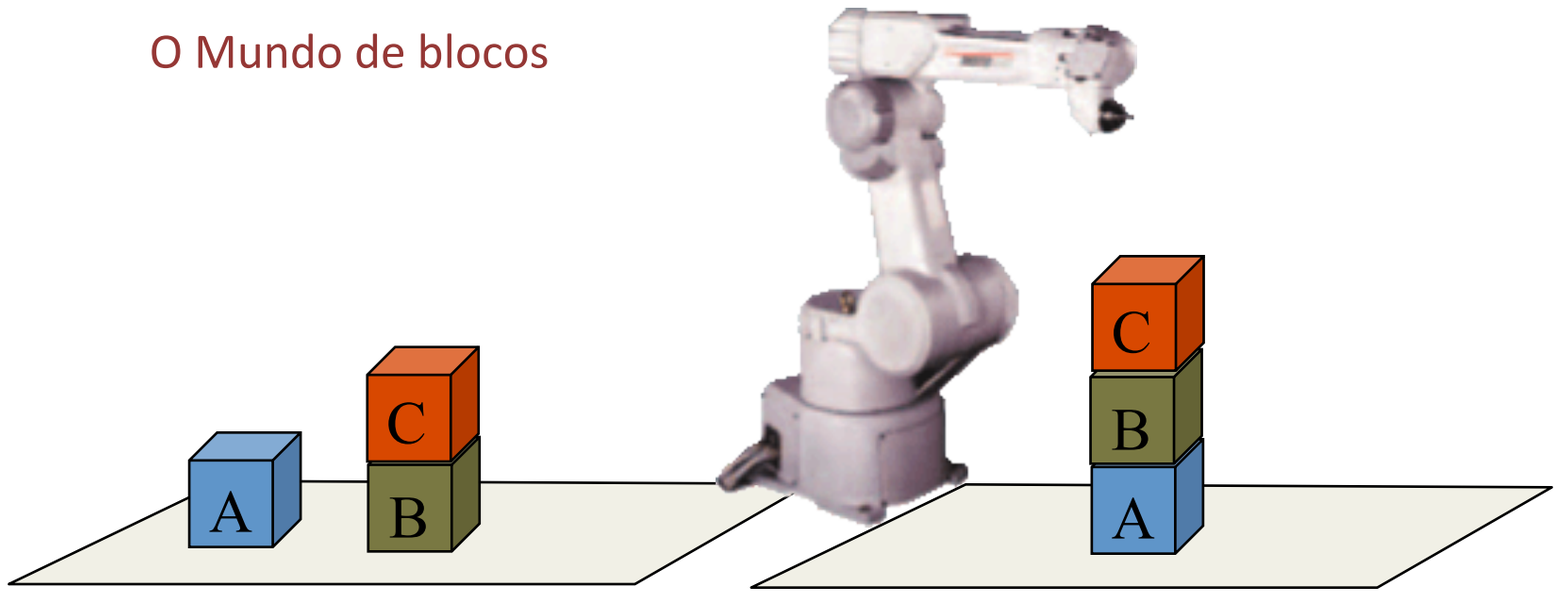
Mathematically, a STRIPS instance is a quadruple , in which each component has the following meaning:

1. is a set of *conditions* (i.e., propositional variables);
2. is a set of *operators* (i.e., actions); each operator is itself a quadruple , each element being a set of conditions. These four sets specify, in order, which conditions must be true for the action to be executable, which ones must be false, which ones are made true by the action and which ones are made false;
3. is the initial state, given as the set of conditions that are initially true (all others are assumed false);
4. is the specification of the goal state; this is given as a pair , which specify which conditions are true and false, respectively, in order for a state to be considered a goal state.

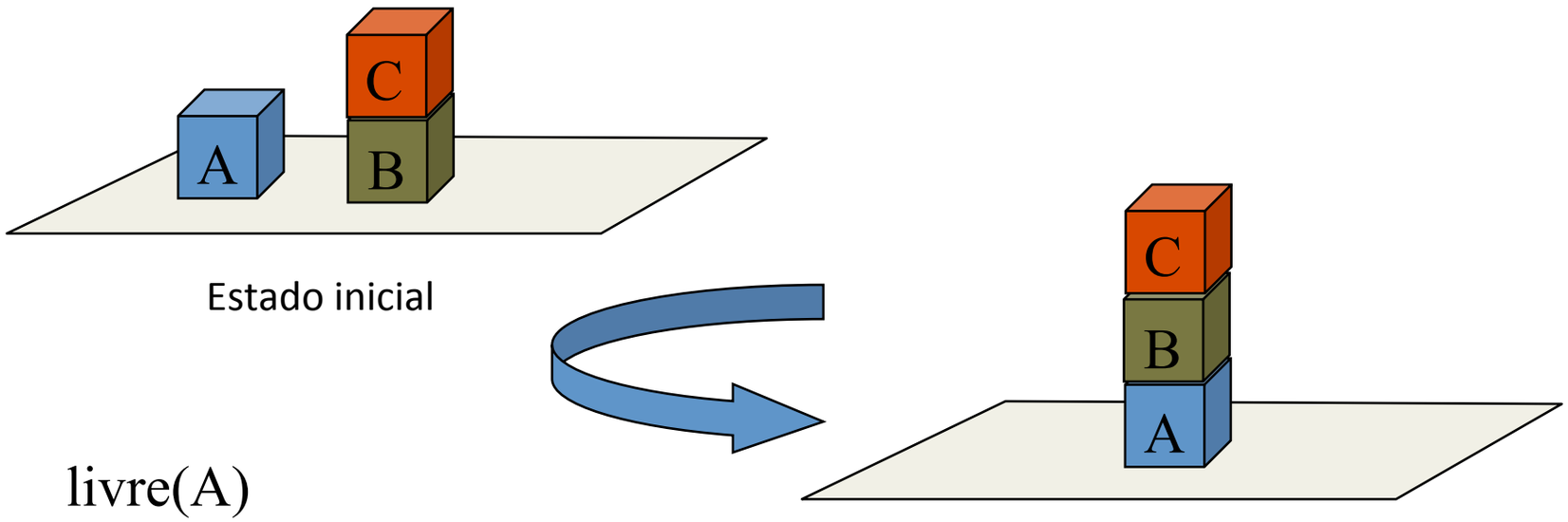
A plan for such a planning instance is a sequence of operators that can be executed from the initial state and that leads to a goal state.

O exemplo mais simples e intuitito do sistema STRIPS é o chamado "mundo de blocos" que consite em mudar blocos de configuração usando robôs.

O Mundo de blocos



Portanto um plano é a solução de um problema composto de um estado inicial, um estado final, e uma sequência de ações (ou um passo) que transforma o estado inicial no estado final, ou, em outras palavras que os coloca na mesma localidade

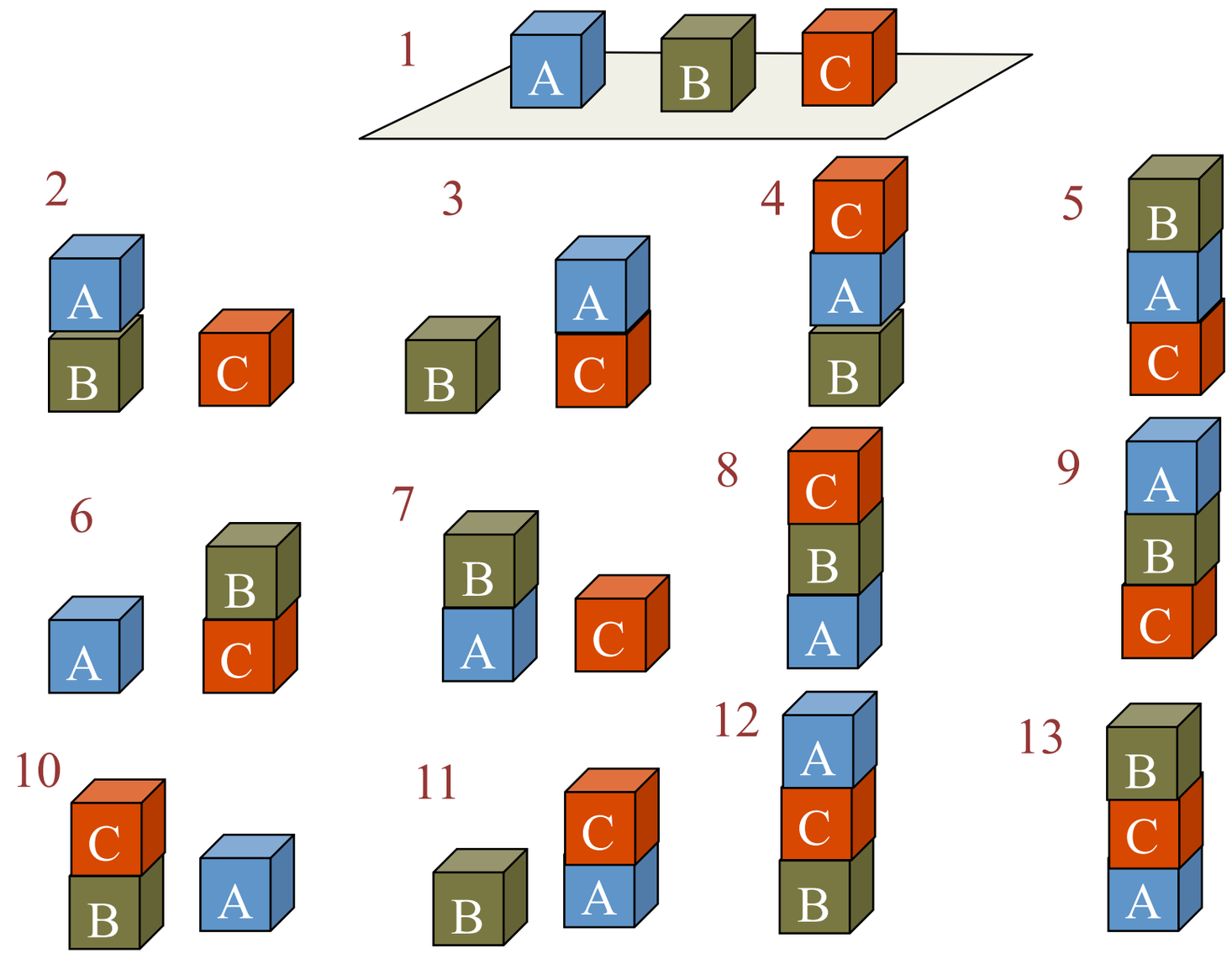


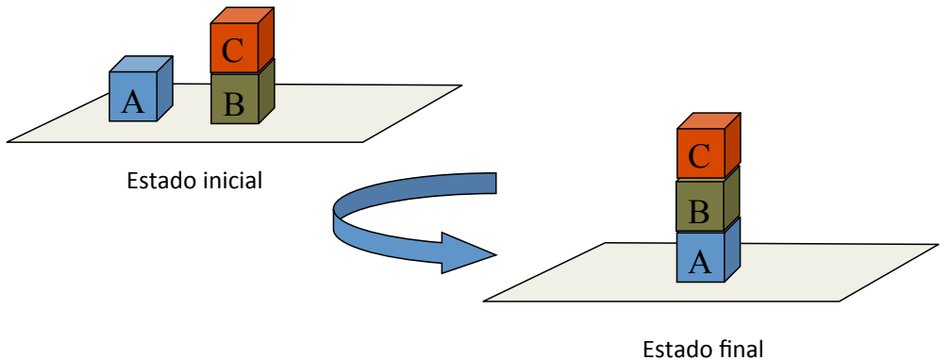
Estado inicial

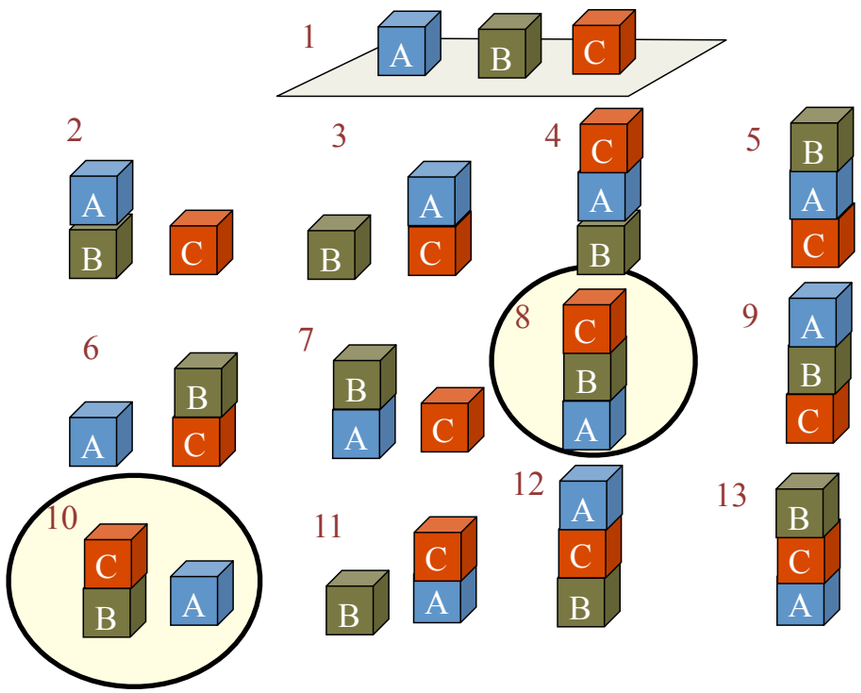
livre(A)
sobre(A,mesa)
sobre(C, B)
sobre (B, mesa)
livre(C)

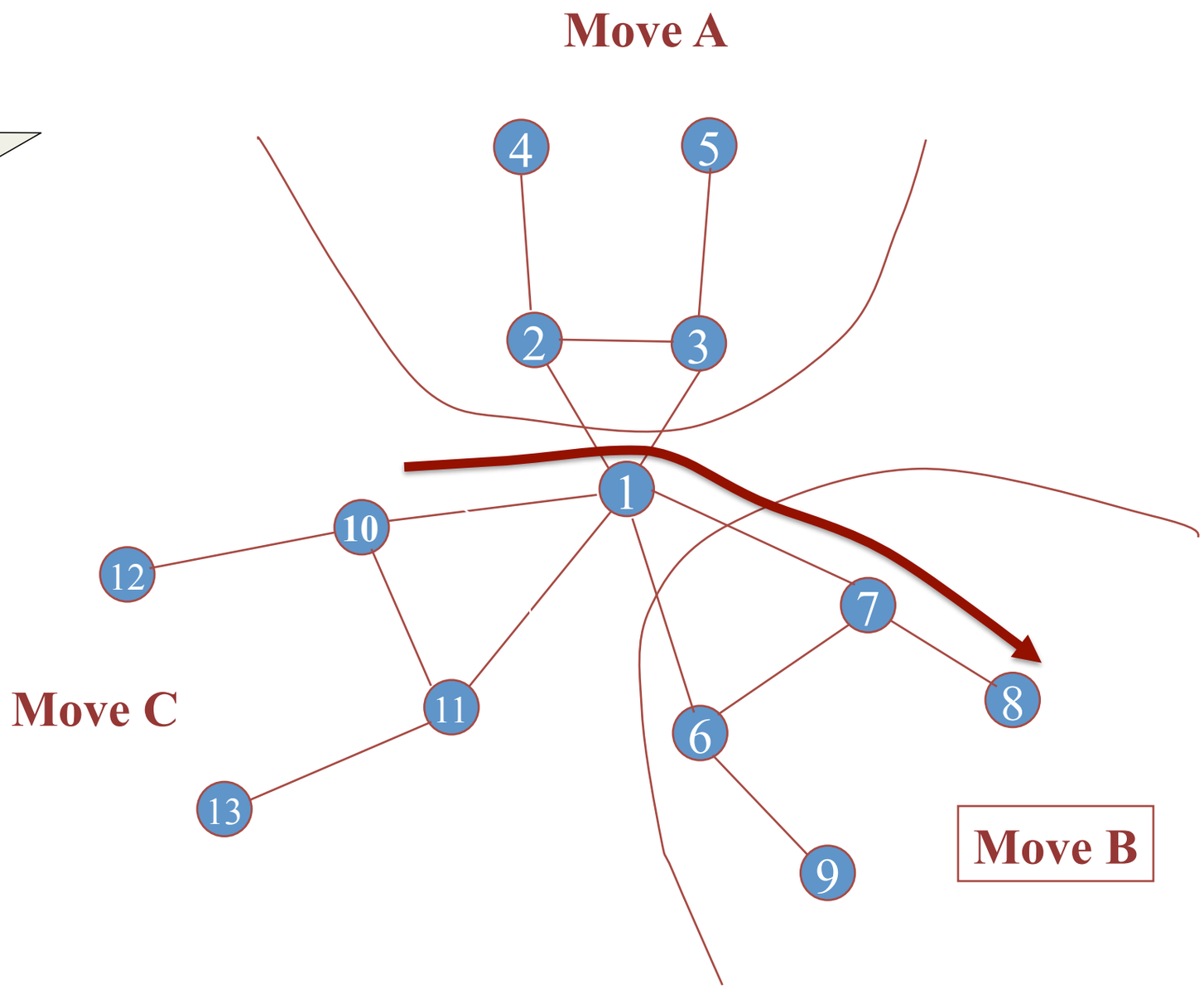
Estado final

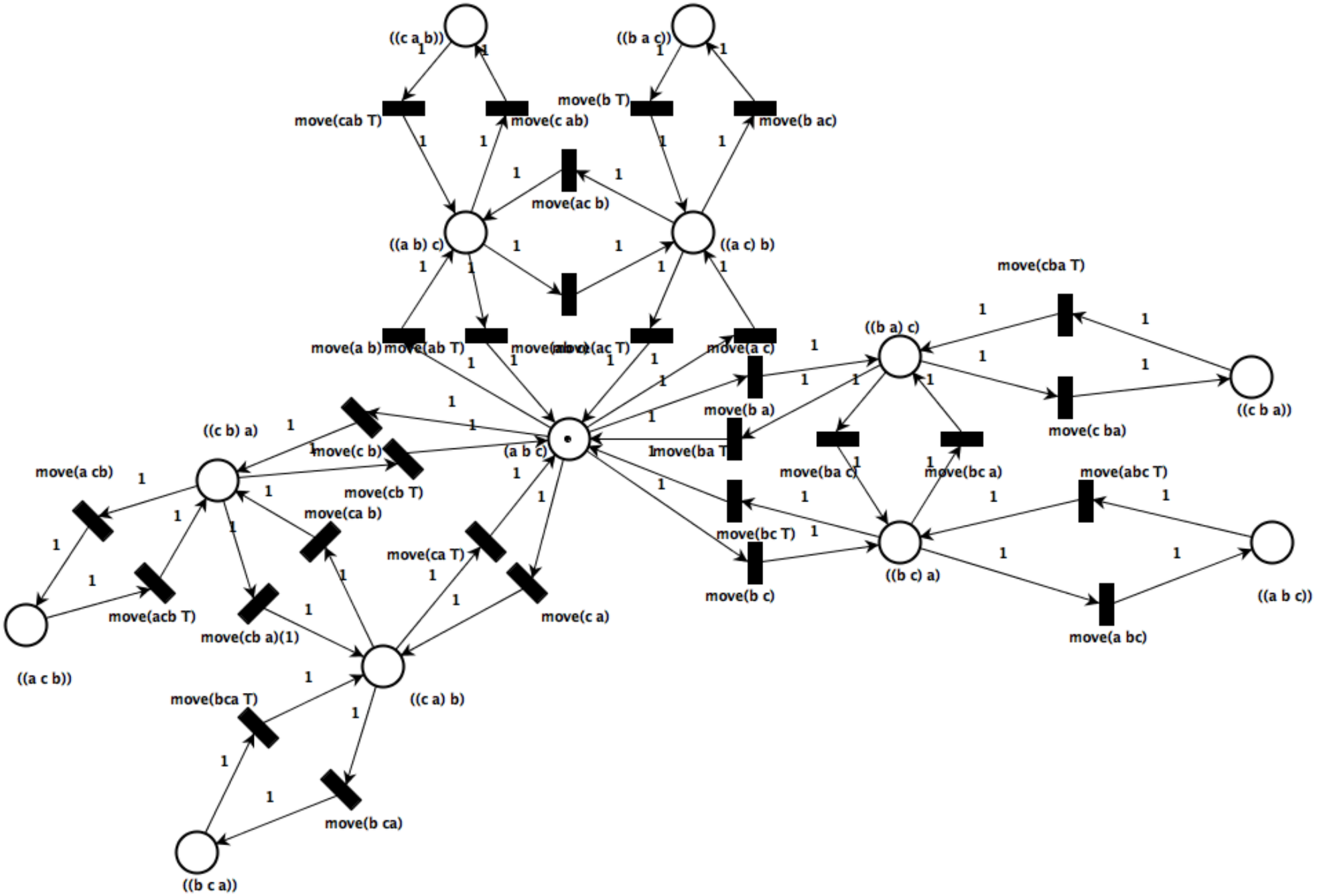
livre(C)
sobre(C,B)
sobre(B,A)
sobre(A,mesa)

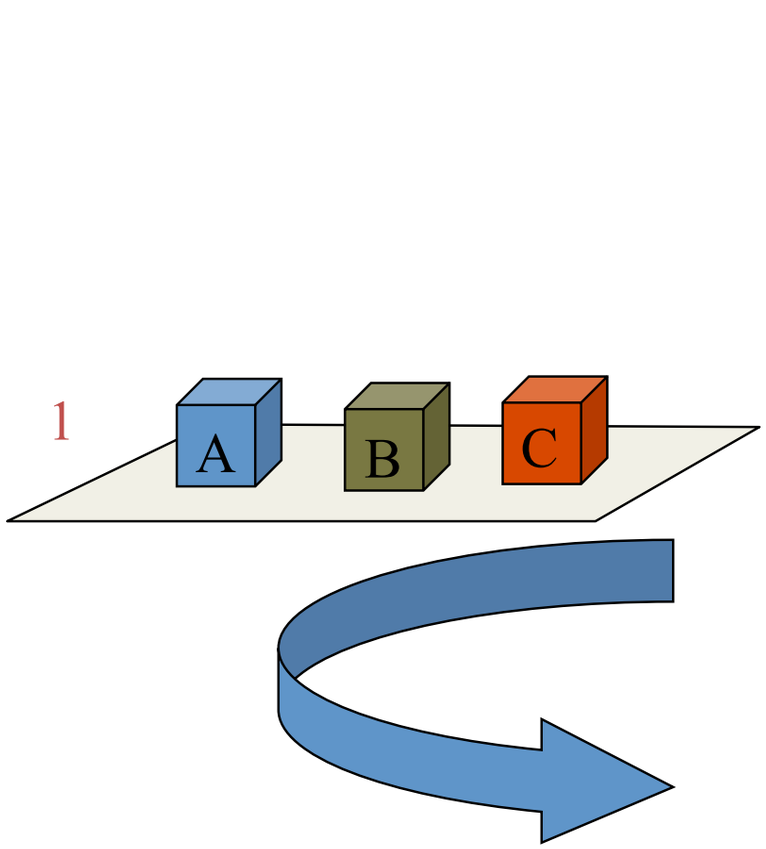

1
A
B
C
2
A
B
C
3
B
A
C
4
C
A
B
5
B
A
C
6
A
B
C
7
B
A
C
8
C
B
A
9
A
B
C
10
C
B
A
11
B
C
A
12
A
C
B
13
B
C
A

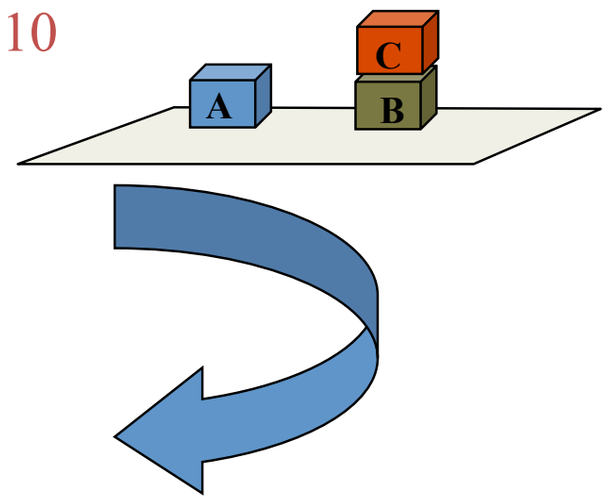

C
A
B
Estado inicial
C
B
A
Estado final

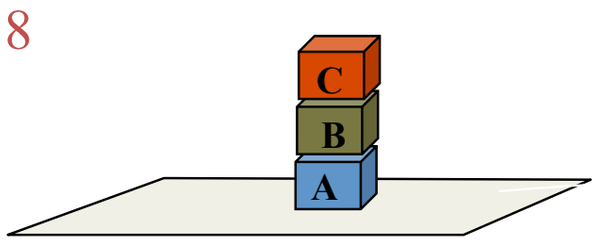

1
A
B
C
2
A
B
C
3
B
A
C
4
C
A
B
5
B
A
C
6
A
B
C
7
B
A
C
8
C
B
A
9
A
B
C
10
C
B
A
11
B
C
A
12
A
C
B
13
B
C
A

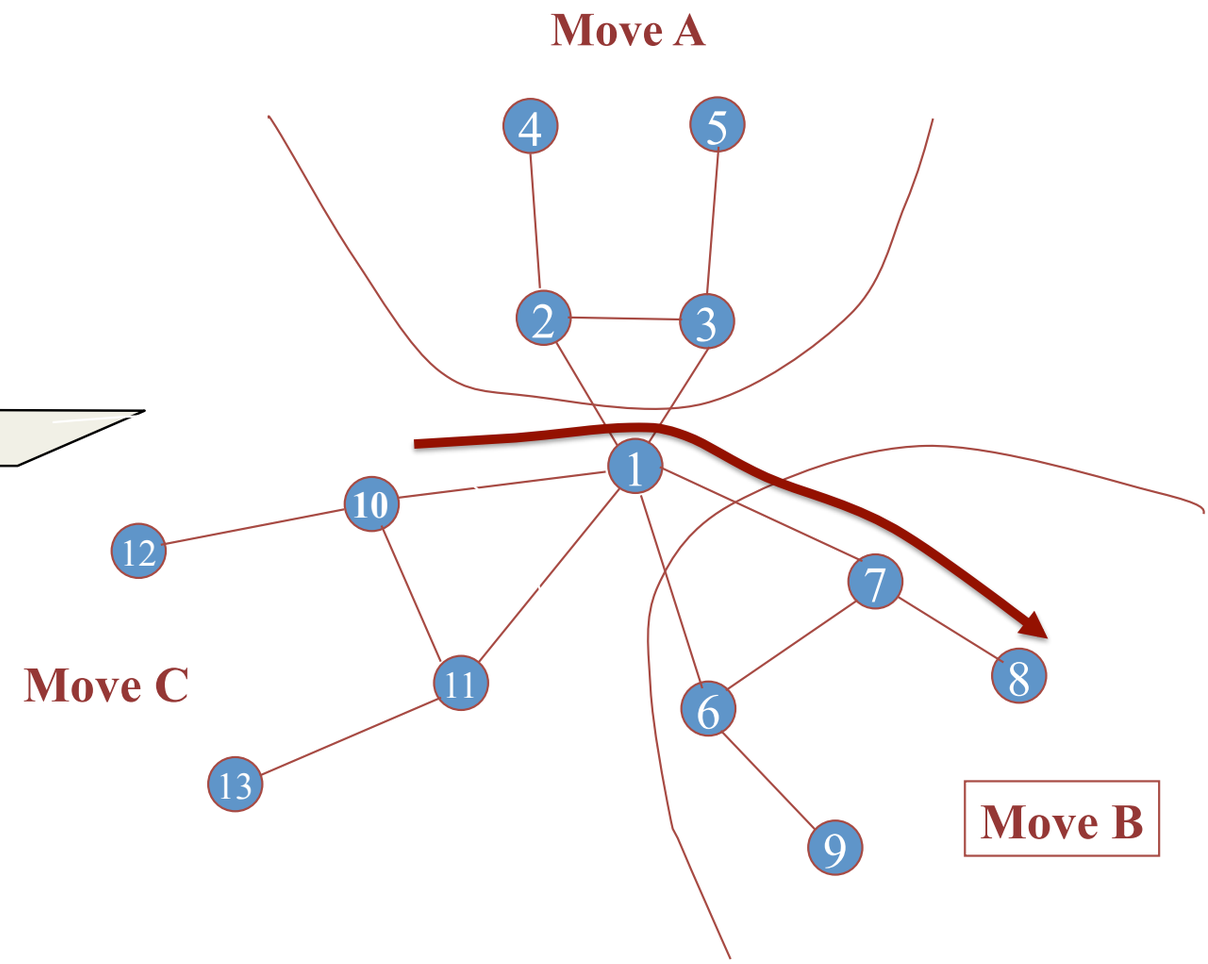

Move A
4
5
2
3
1
10
12
7
Move C
11
8
6
13
9
Move B

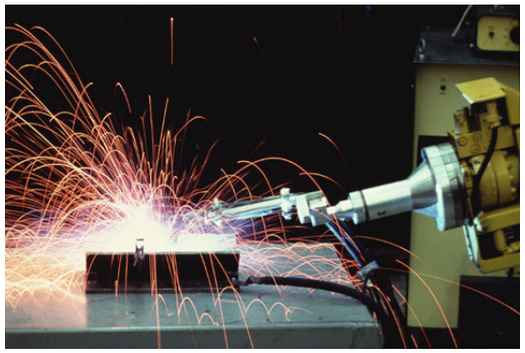

((c a b))
((b a c))
move(cab T)
move(c ab)
move(b T)
move(b ac)
move(ac b)
((a b) c)
((a c) b)
move(cba T)
move(a b)
move(ab T)
move(ac T)
move(a c)
((b a) c)
move(b a)
((c b a))
((c b) a)
move(c b)
(a b c)
move(ba T)
move(c ba)
move(a cb)
move(cb T)
move(ba c)
move(bc a)
move(abc T)
move(ca b)
move(bc T)
move(ca T)
((b c) a)
((a b c))
move(acb T)
move(cb a)(1)
move(c a)
move(b c)
move(a bc)
((a c b))
move(bca T)
((c a) b)
move(b ca)
((b c a))


1
A
B
C


10
A
C
B


8
C
B
A

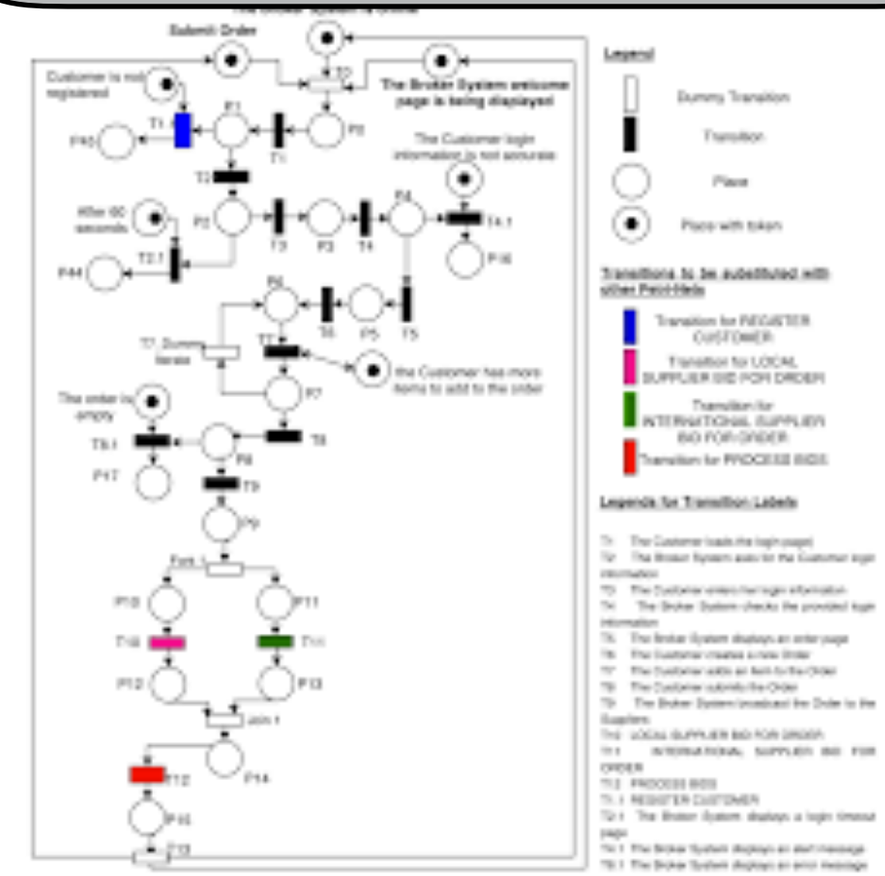

Move A
4
5
2
3
1
10
12
7
8
Move C
11
6
13
9
Move B



GA ⟺ **TG**

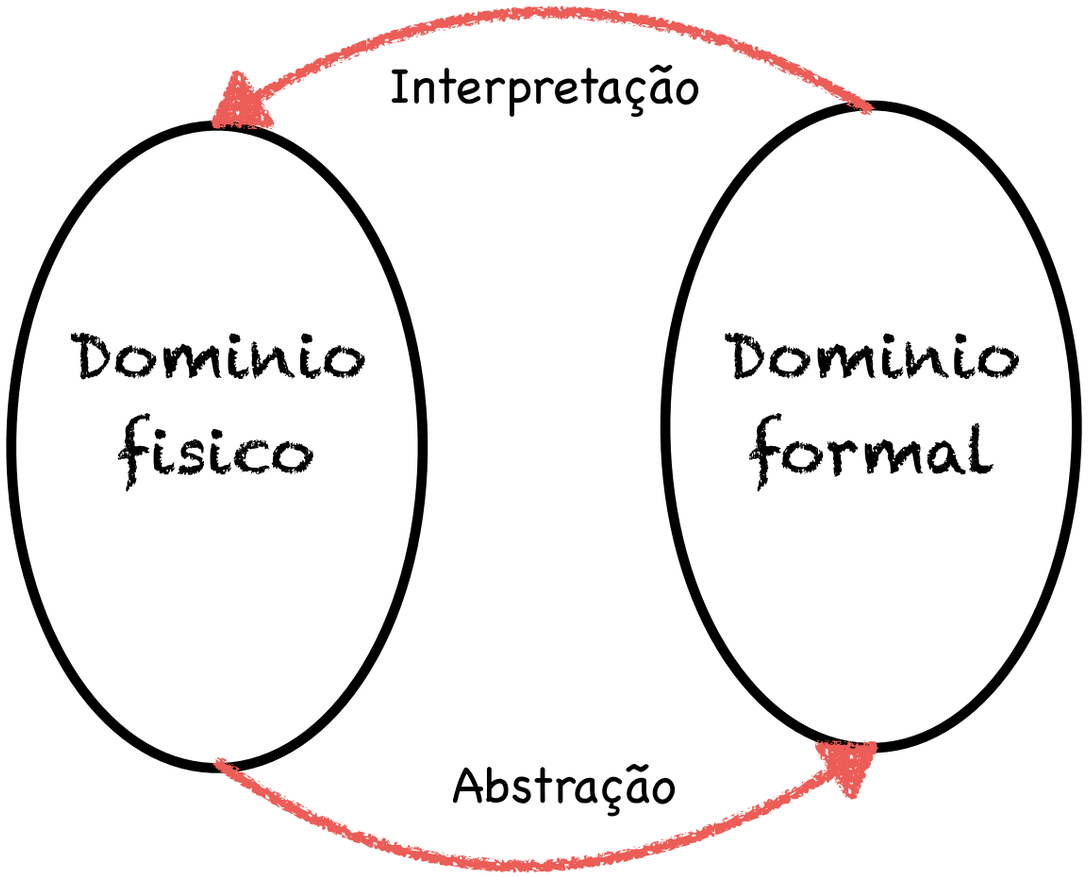
Seqüenciamento de tarefas (planning)

- trajetórias de AGV´s
- montagem
- problemas modelo (mundo de blocos)
- planejamento reativo



Novas aplicações

- Redes de computadores
- Sistemas logísticos
- Engenharia de software
- Análise de requisitos
- Sistemas de transporte
- Sistemas biológicos
- Análise de workflow
- …



Até aqui vimos como "modelar" um sistema de pequeno porte usando redes elementares e a intuição somente. Portanto o que fizemos foi uma abstração, à partir de uma análise de causa-efeito o mundo fisico.

O que mais pode ser feito?

Fim