

PMR 5237

Modelagem e Design de Sistemas Discretos em Redes de Petri

Aula 8: Redes Coloridas

Prof. José Reinaldo Silva
reinaldo@usp.br

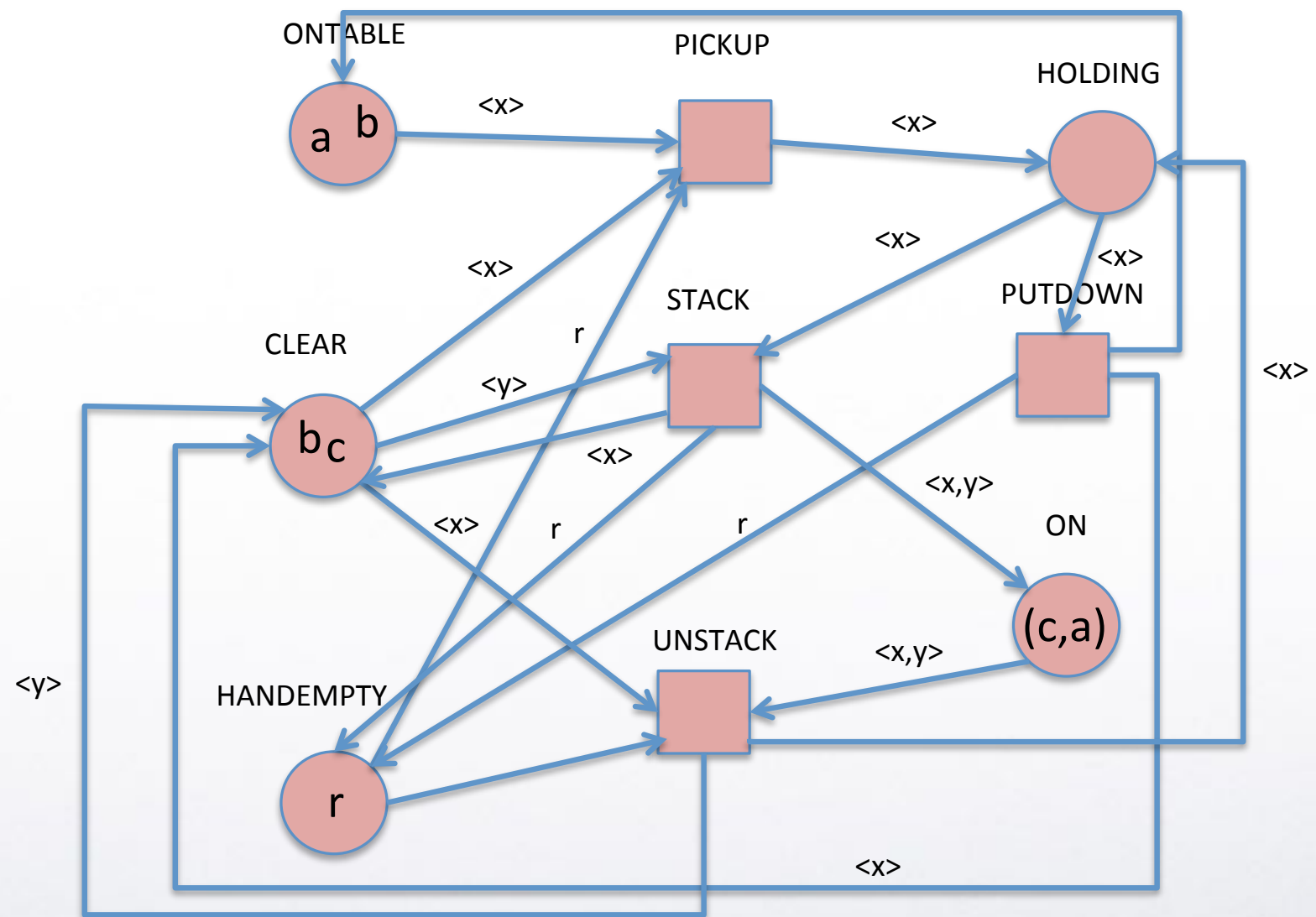
O problema (de planejamento)

Problema já está modelado e com uma representação já definida para especificação de estados, e uma série de ações que modificam estes estados. Estes estados atuam sobre um conjunto de blocos individualizados (distinguíveis). Portanto a modelagem deste tipo de problema é direta e deve ser feita com vantagem usando a HLPN (comparado a fazer a rede clássica primeiro).

A modelagem do problema fica assim bem simples.

$\text{Blk} := \{a, b, c\}$
 $R := \{r\}$
 $\text{OnB: Blk} \times \text{Blk}$

$x, y: \text{Blk}$



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

Tente fazer isso, mesmo já tendo visto a solução.

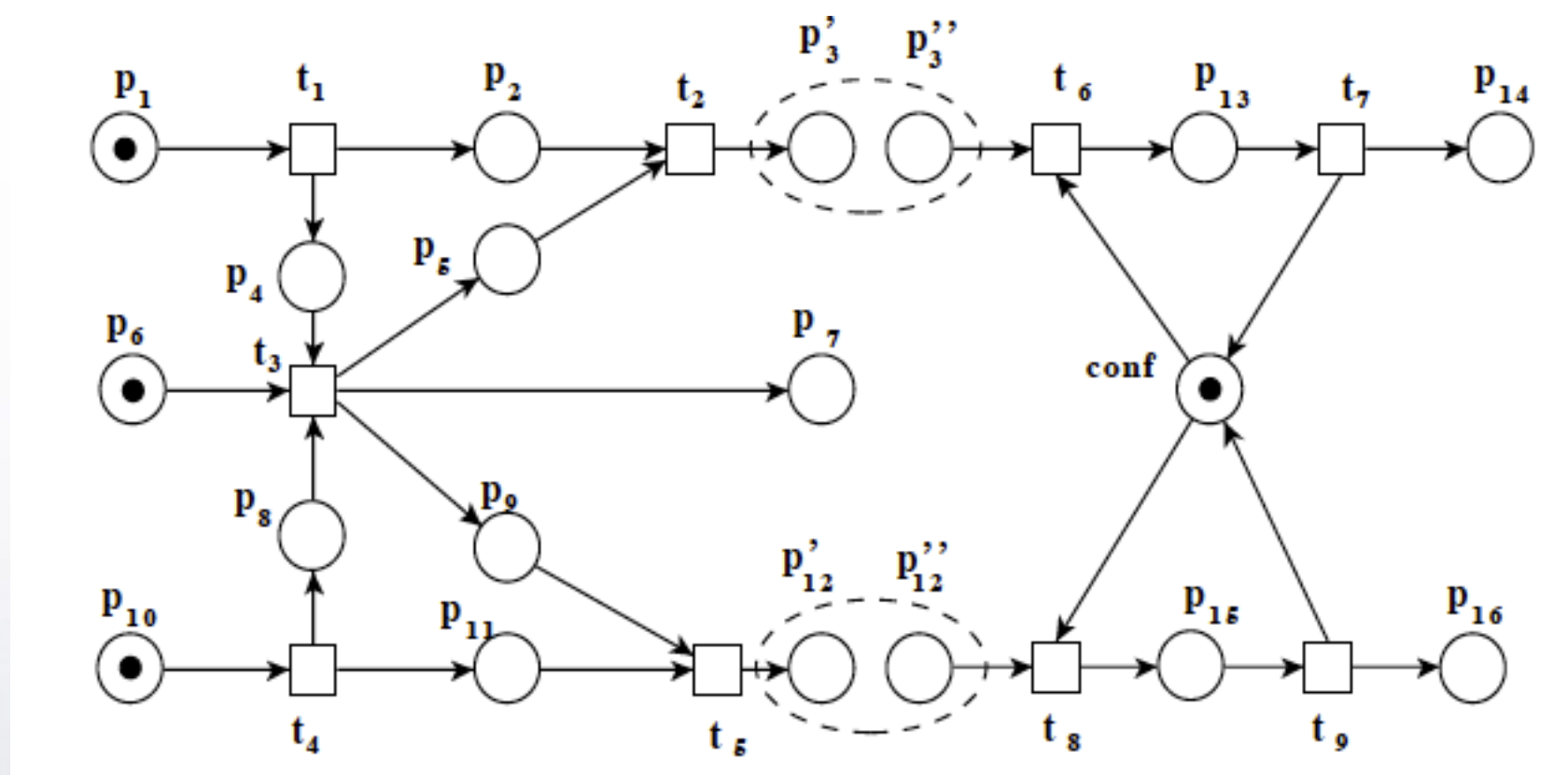
Coloured Petri Nets

Definition 4.3.1. A coloured Petri net (CPN) is defined by a tuple $\mathcal{N} = \langle P, T, \text{Pre}, \text{Post}, \mathcal{C}, cd \rangle$, where

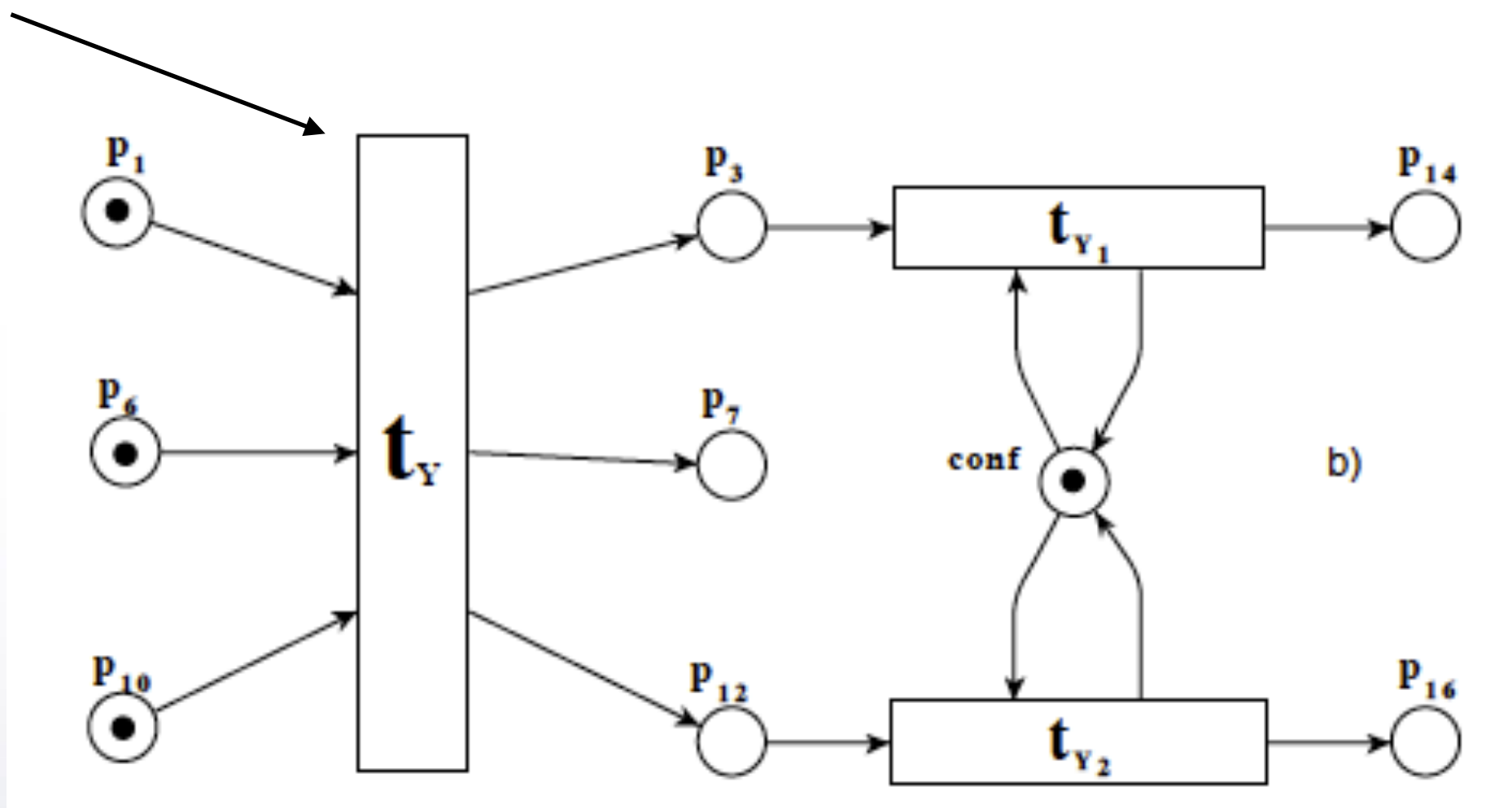
- P is a finite set (the set of places of $\mathcal{N}$),
- T is a finite set (the set of transitions of $\mathcal{N}$), disjoint from P ,
- $\mathcal{C}$ is the set of colour classes,
- $cd: P \cup T \rightarrow \mathcal{C}$ is the colour domain mapping, and
- $\text{Pre}, \text{Post} \in \mathcal{B}^{|P| \times |T|}$ are matrices (the backward and forward incidence matrices of $\mathcal{N}$) such that $\text{Pre}[p, t] : cd(t) \rightarrow \text{Bag}(cd(p))$ and $\text{Post}[p, t] : cd(t) \rightarrow \text{Bag}(cd(p))$ are mappings for each pair $(p, t) \in P \times T$.

$\mathcal{B}$ can be taken as the set of mappings of the form $f : cd(t) \rightarrow \text{Bag}(cd(p))$. Again, $\mathbf{C} = \text{Post} - \text{Pre}$ is called the incidence matrix.

Vamos considerar novamente o problema da largada de fórmula 1, agora seguido de um checkpoint onde se verifica a posição de cada carro.



macrotransição

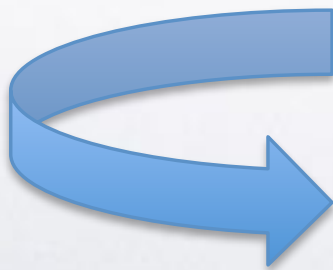


É possível dobrar esta rede? O que fazer como o dobramento?

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

Definição informal das CPNs

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.

Aplicações

Aplicações Típicas

Protocolo de Comunicação

Redes de Dados

Algoritmos Distribuídos

Sistemas Embarcados

Novas Aplicações

Sistemas de workflow

Sistemas de manufatura

Sistemas multi-agente

Processos de negócio

Análise de Requisitos



At a time when it is increasingly understood that programs must withstand rigorous analysis, particularly for systems where safety is critical, a rigorous language presentation is even important for negotiators and contractors; for a robust program written in an insecure language is like a house built upon sand.

Robin Milner

As redes coloridas demarcam uma fase onde se buscava não apenas enfrentar a “explosão combinatória”, do espaço de estados mas garantir o formalismo da modelagem.

Portanto as redes coloridas (e de alto nível) devem ser vistas como um convite a aumentar o nível de abstração dos processos de modelagem em Engenharia.

Redes Coloridas (desfazendo os mitos)

As vantagens das redes coloridas seriam:

1. Explorar as simetrias com o objetivo de reduzir o tamanho da rede.

Falha: o grafo é reduzido mas a informação para para as inscrições a complexidade da representação não se altera significativamente

2. Ter uma representação abstrata para a rede clássica.

Falha: verdade em princípio, mas ter que reduzir a representação para rede clássica cada vez que deseja fazer análise de propriedades pode não ser exatamente uma vantagem.

3. Aumentar o poder de expressão da representação baseada em redes.

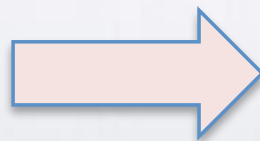
Falha: simplesmente não é verdade.

Uso prático das redes CPN

Como no caso das redes clássicas o uso prático das redes CPN está associado a **Simulação** do modelo, e portanto ao estudo de cenários específicos e ao processo de evolução das marcas.

Temos portanto o mesmo problema de desenvolver métodos alternativos de análise baseados nas propriedades da rede e fugir do problema da atingibilidade.

Novos métodos



Verificação e Model checking

Processos especiais

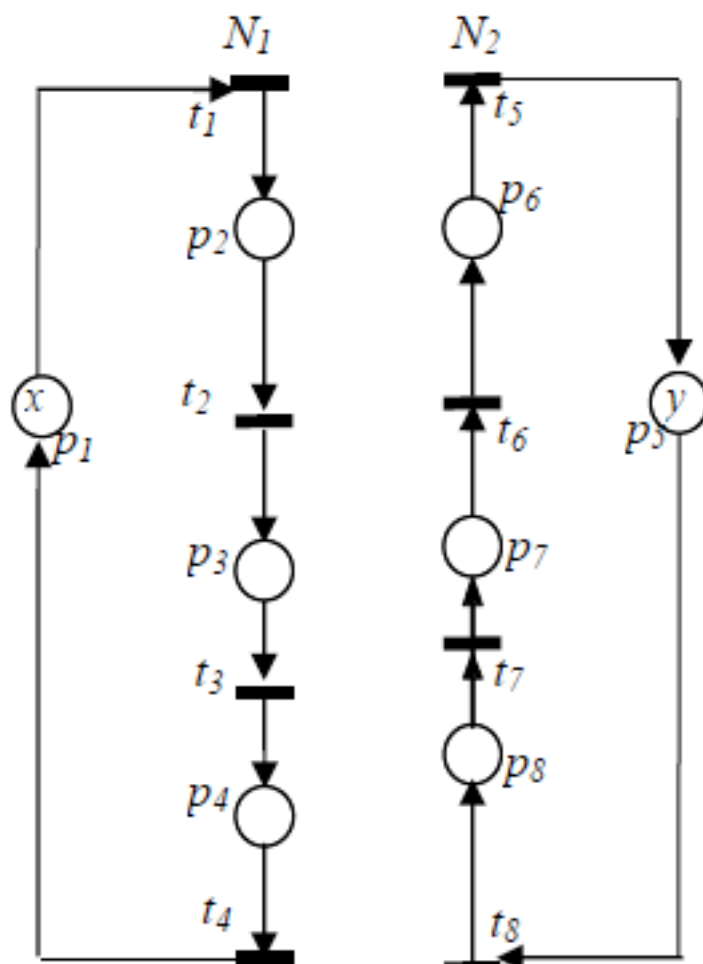


Fig. 1(b) Two S^2P .

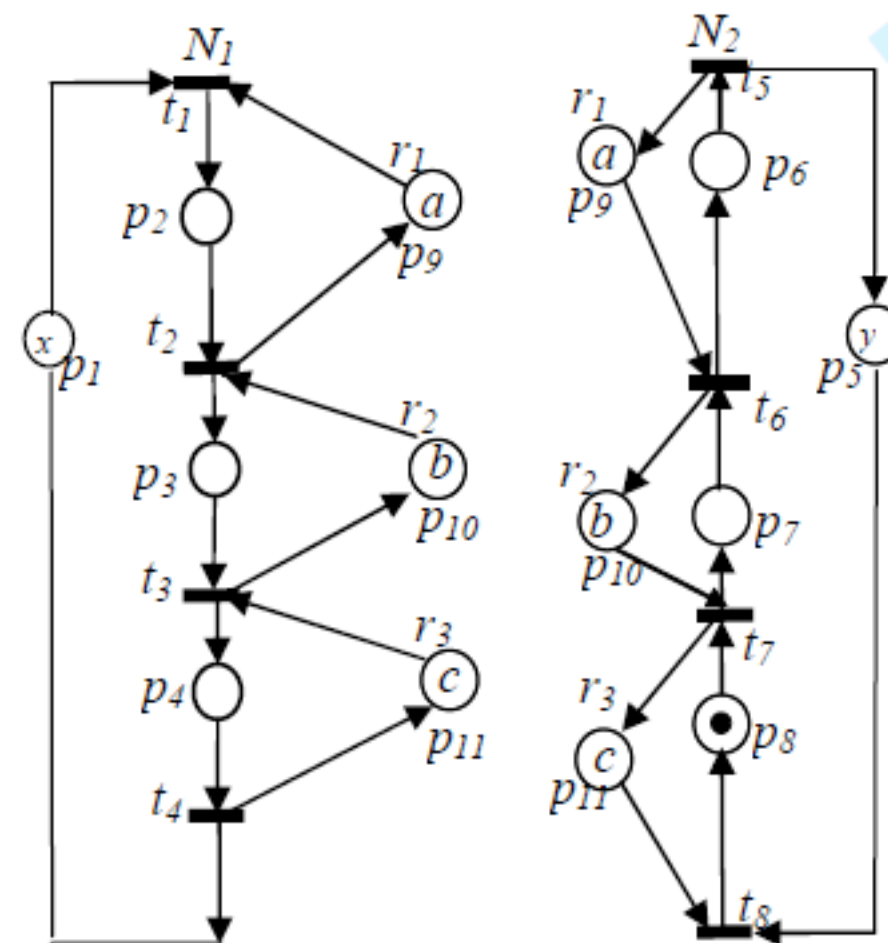


Fig. 1(c) S^2PR of N_1 and N_2

O problema do acoplamento com recursos

O sistema S2PR pode ainda ser acoplado de modo que os processos sequenciais interferem um no outro podendo causar atrasos e até deadlocks devido à falta de recursos no momento devido.

Li, Z.W. and Zhou, M.C., "Deadlock resolution in automated manufacturing systems: A novel Petri net approach," Springer, London, 2009

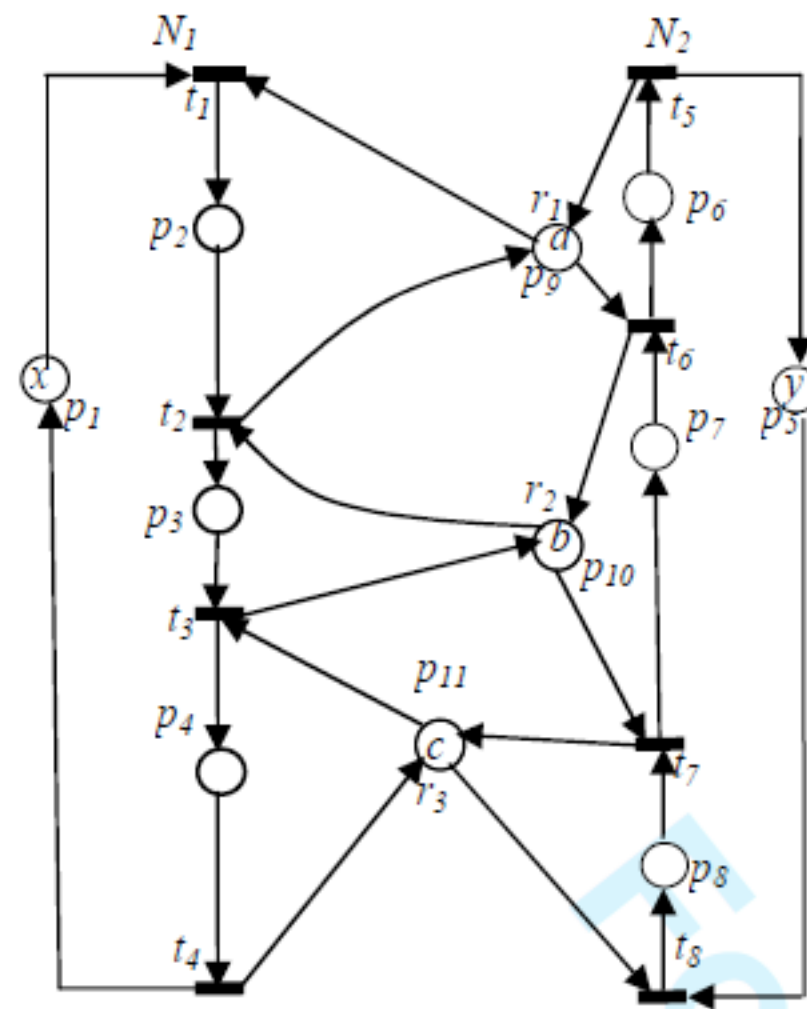


Fig. 1(a) S^3PR & 3rd-order system ;
 $a=b=c=1$.

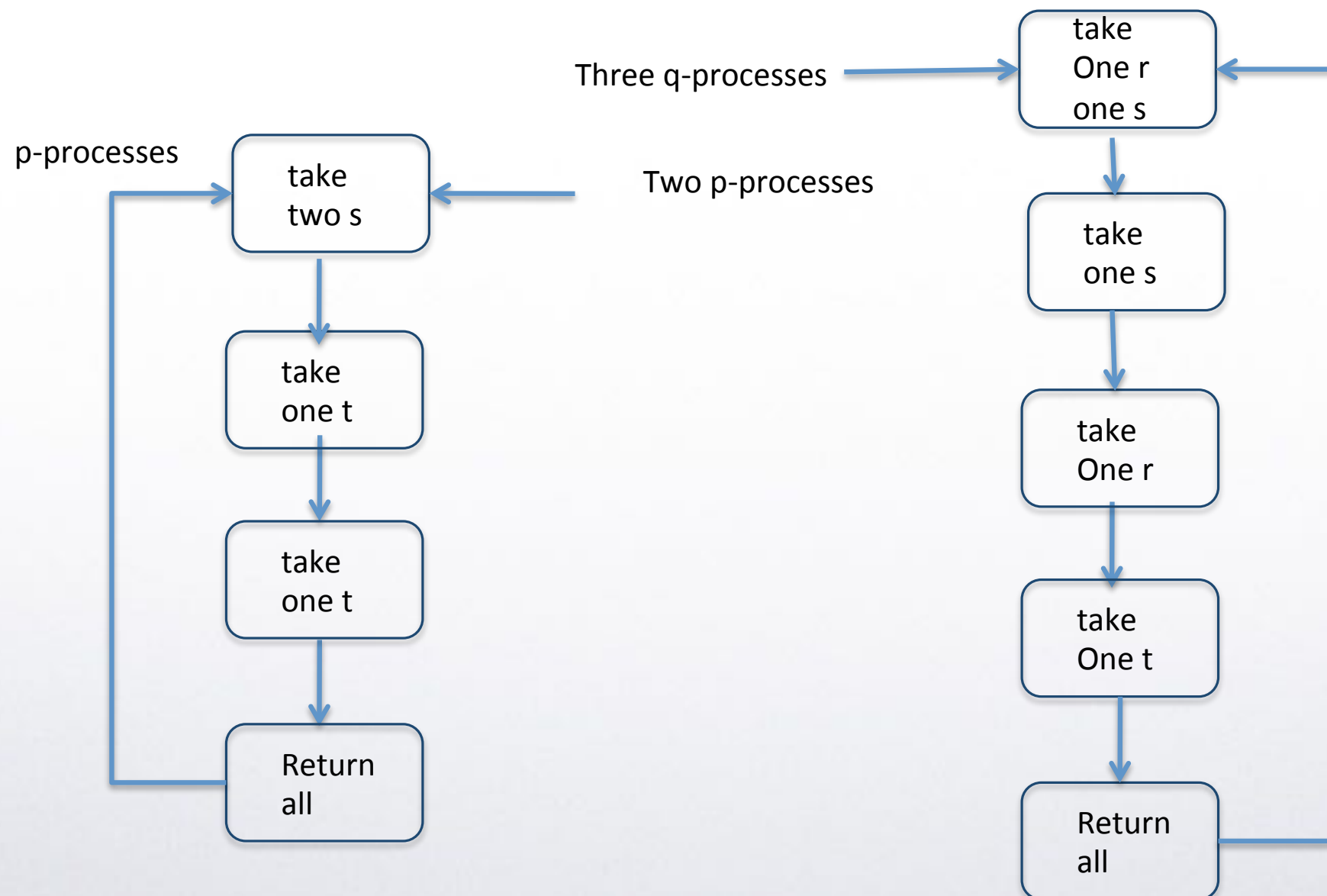
Introdução informal: O problema da alocação de recursos



Na indústria automotiva moderna é comum se ter vários processos ou linhas de montagem e nestes um ou mais tipos de automóvel sendo montados em pipeline. Isso traz um problema, que é ter o tipo certo de insumo ou recurso no tempo correto, para a matriz correta.

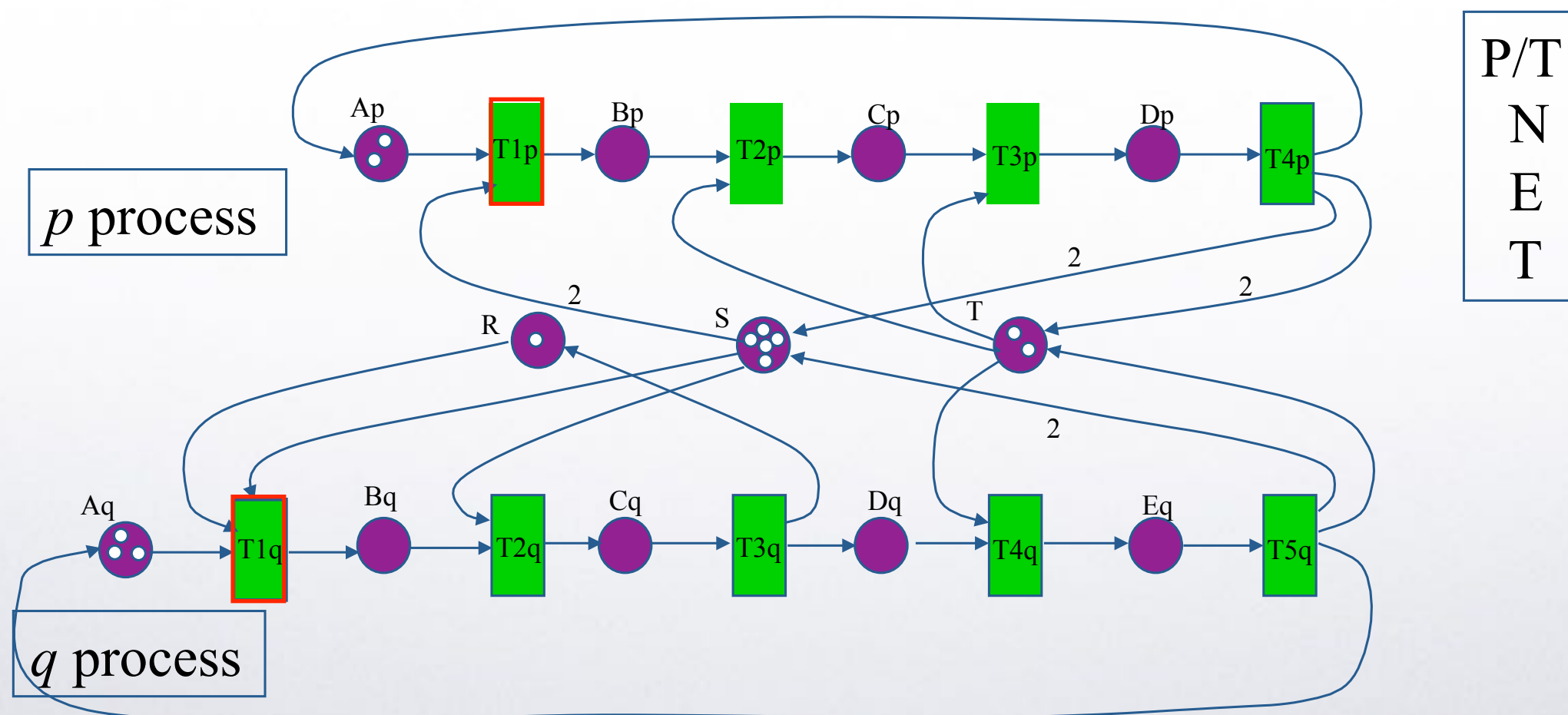
Um problema similar e igualmente importante é ter processos homogêneos mas compartilhando o mesmo centro de recursos.

Estruturação do problema

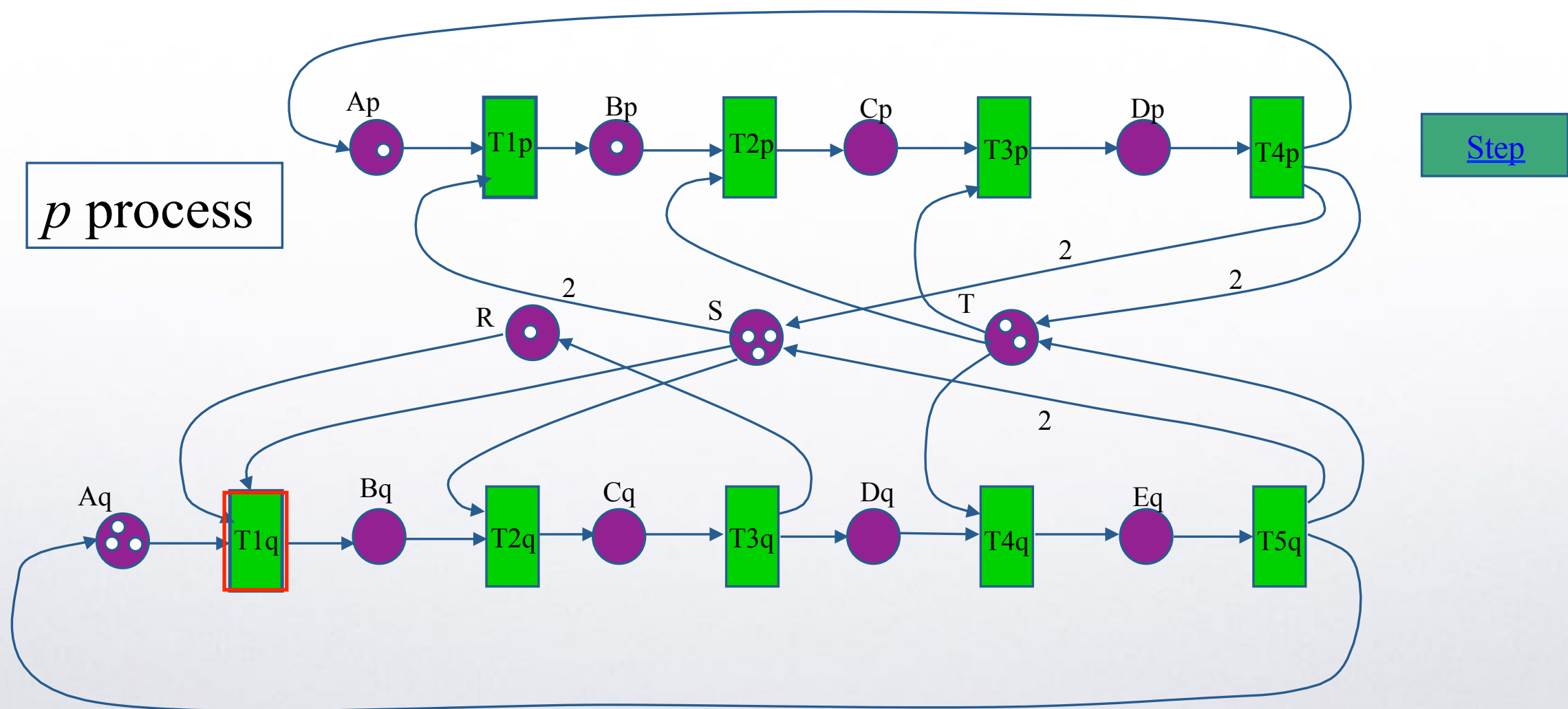


Modelagem clássica

Seja o seguinte sistema de alocação de recursos, caracterizado por dois processos concorrentes,

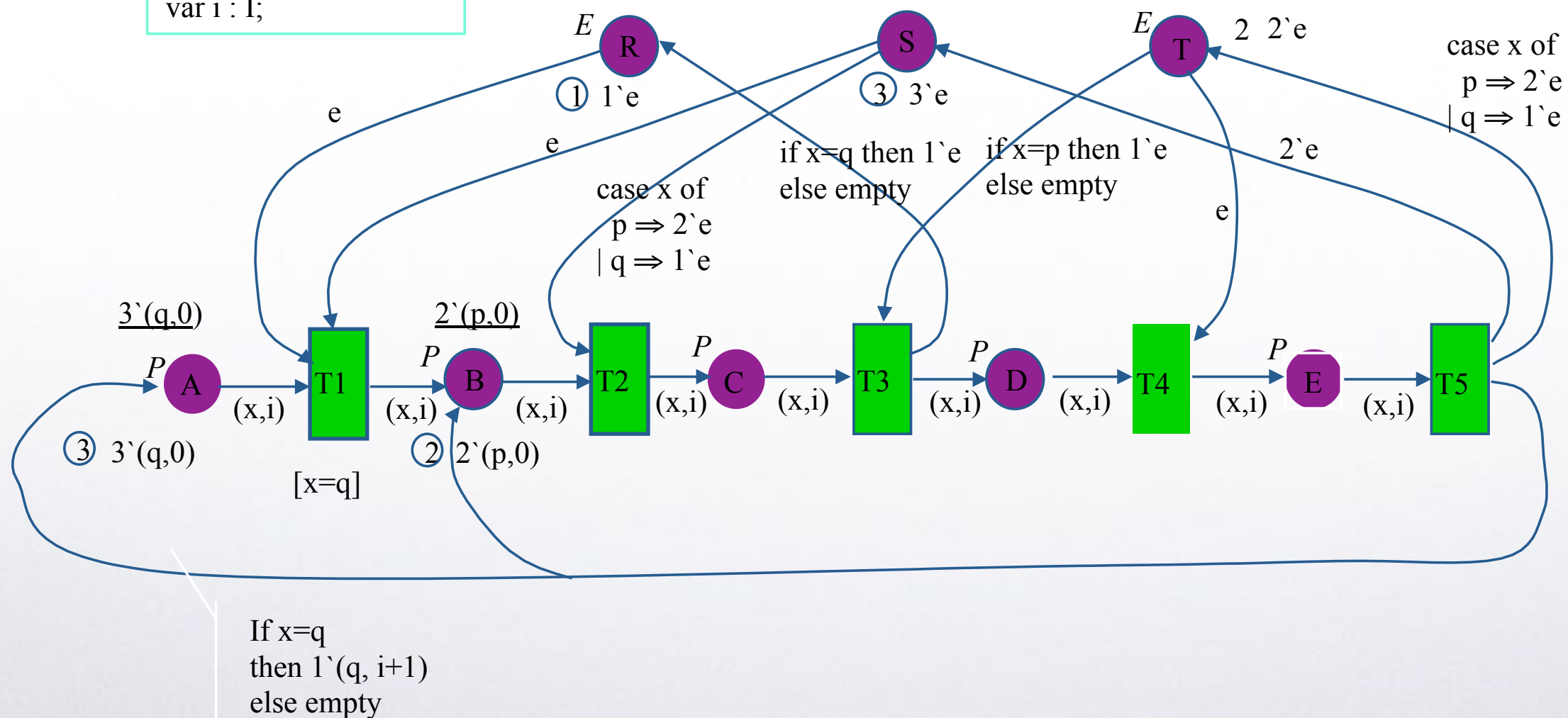


Passos e multisetes



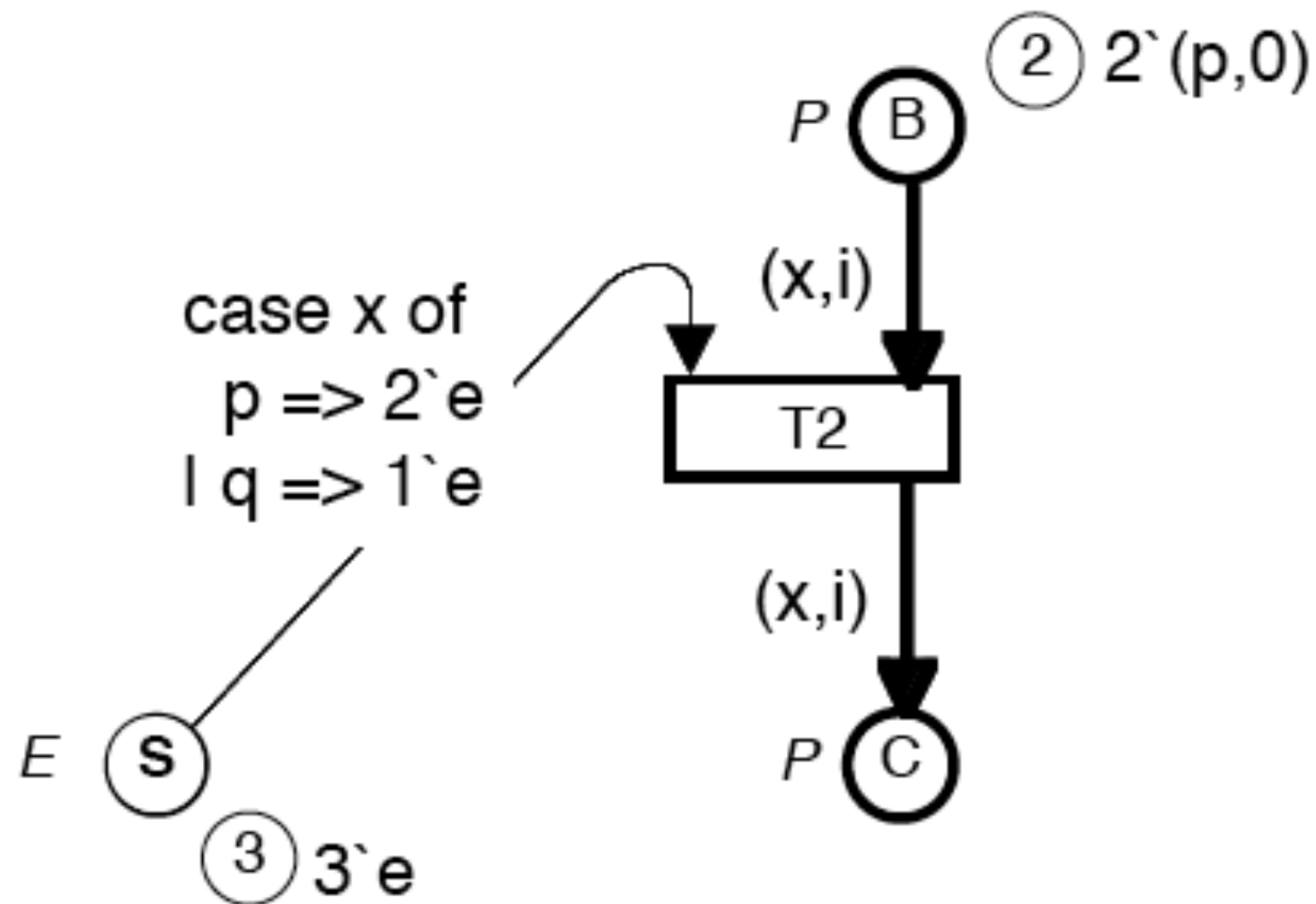
Para sintetizar uma rede colorida devemos explorar a simetria entre os processos, isto é, produzir um único processo que representa tanto o processo p como o processo q . Neste caso as marcas passam a ser distinguíveis para representar os dois processos.

color U=with p | q;
 color I = int;
 color P = product U*I;
 color E= with e;
 var x : U;
 var i : I;

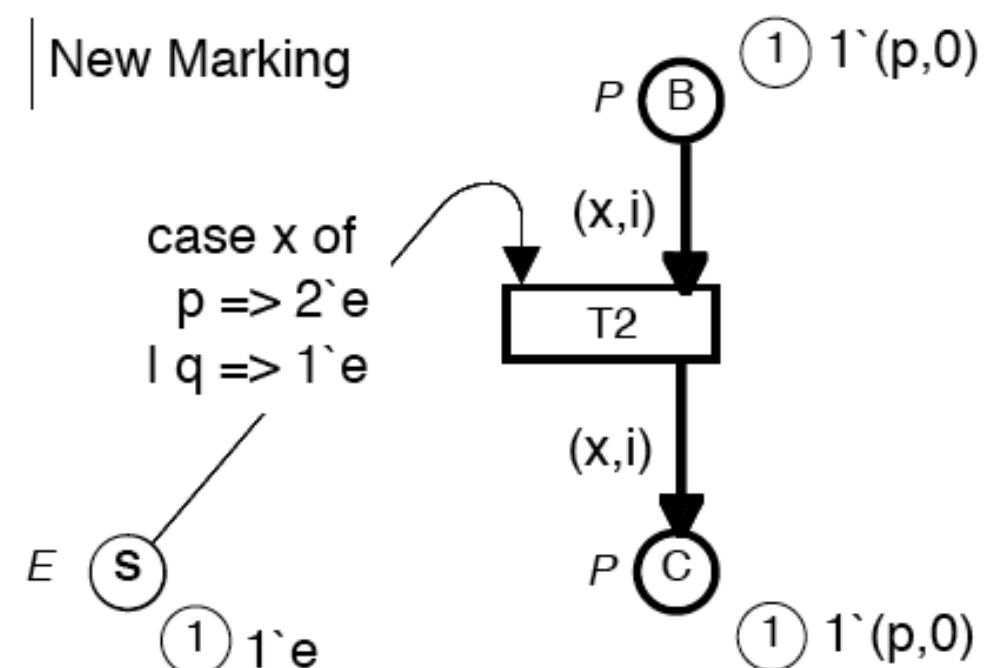
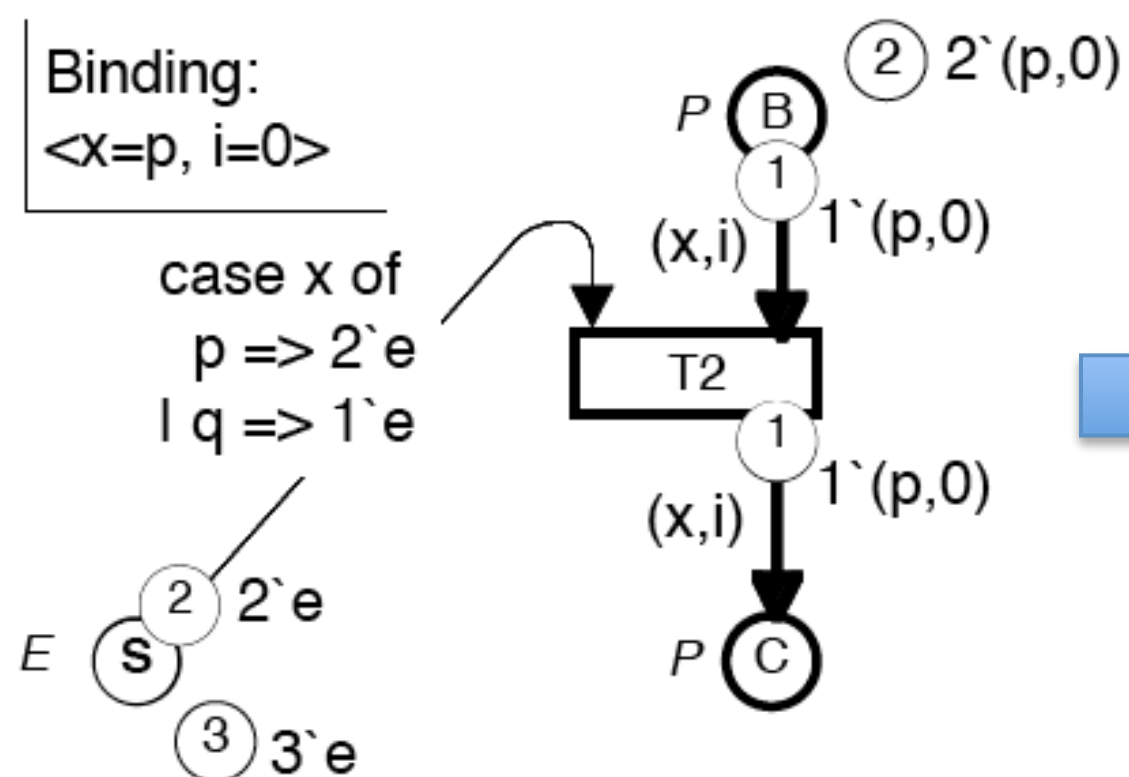


CPN
 NET

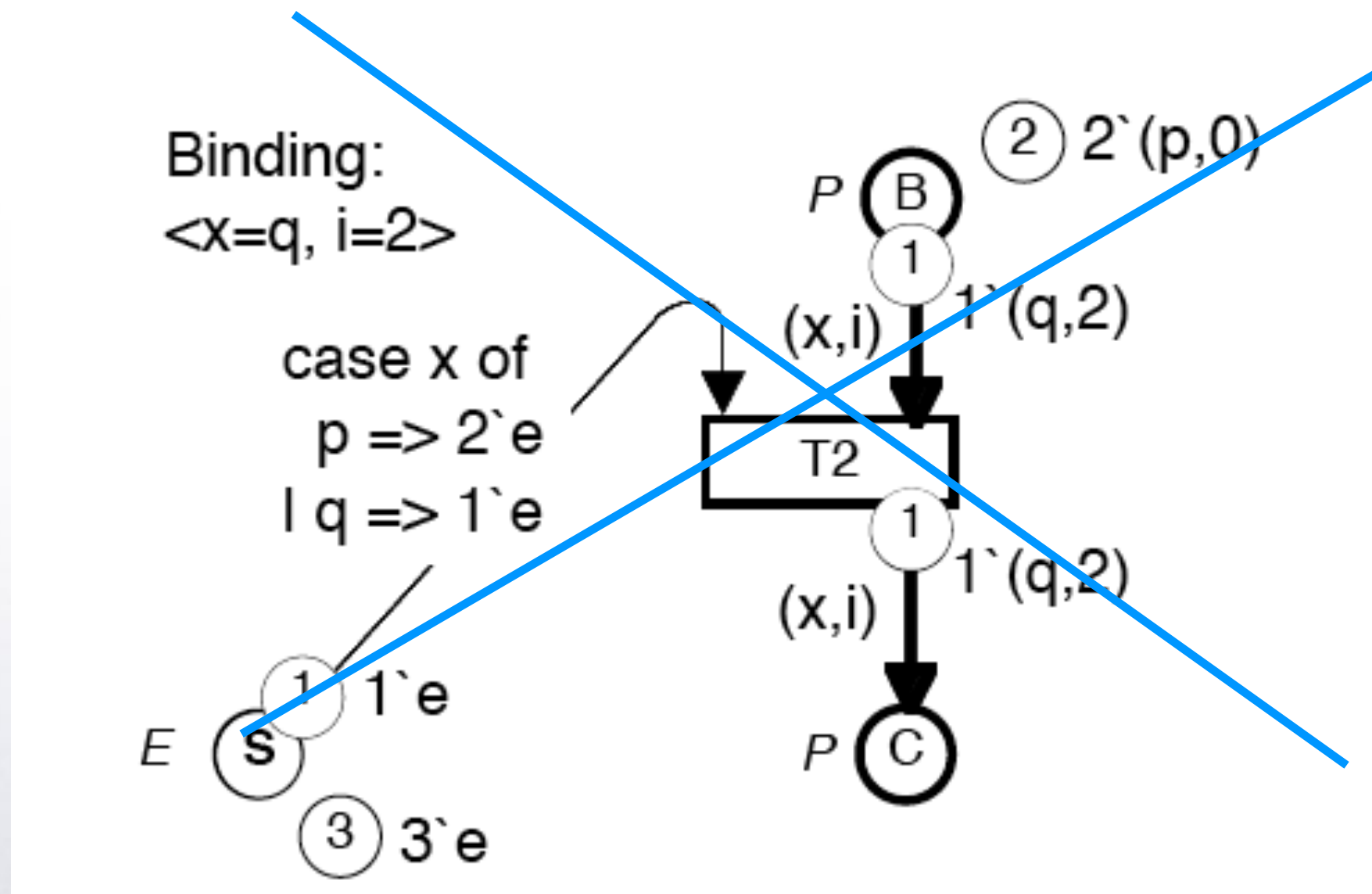
Biding



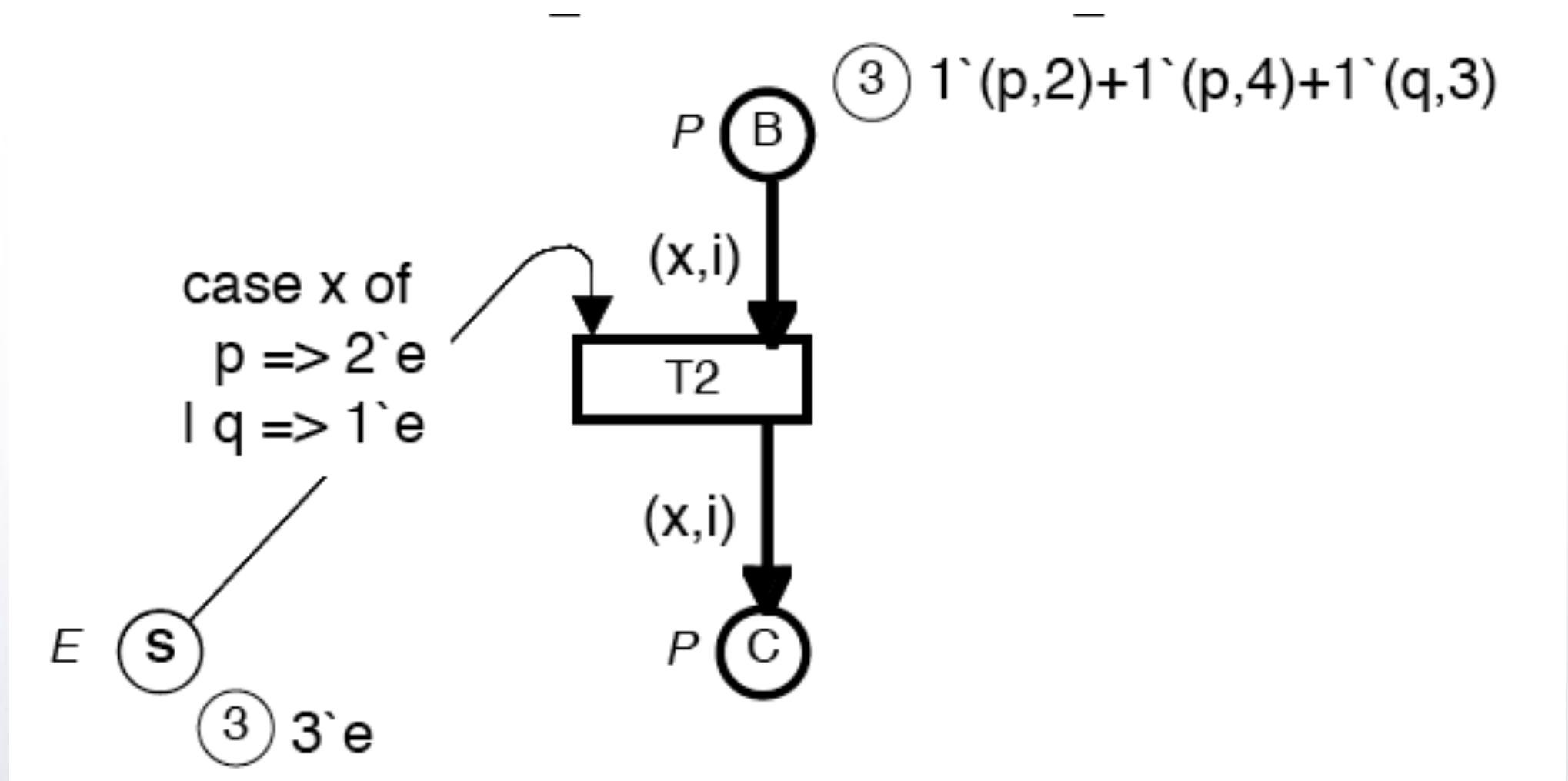
Disparo na rede



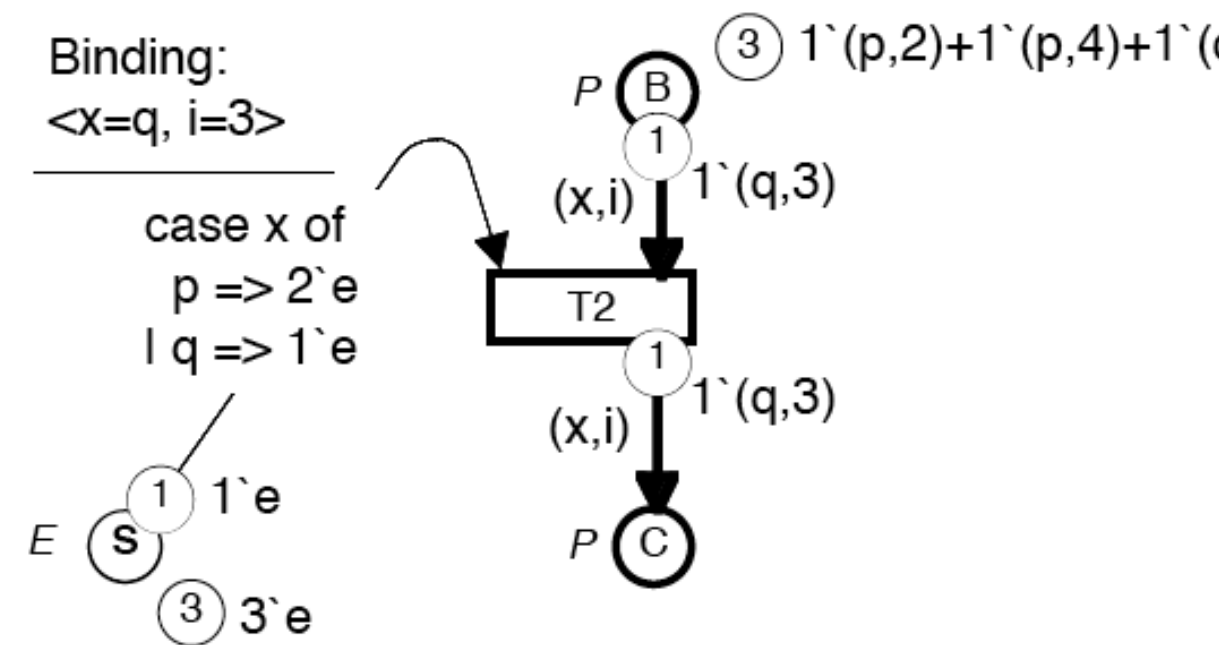
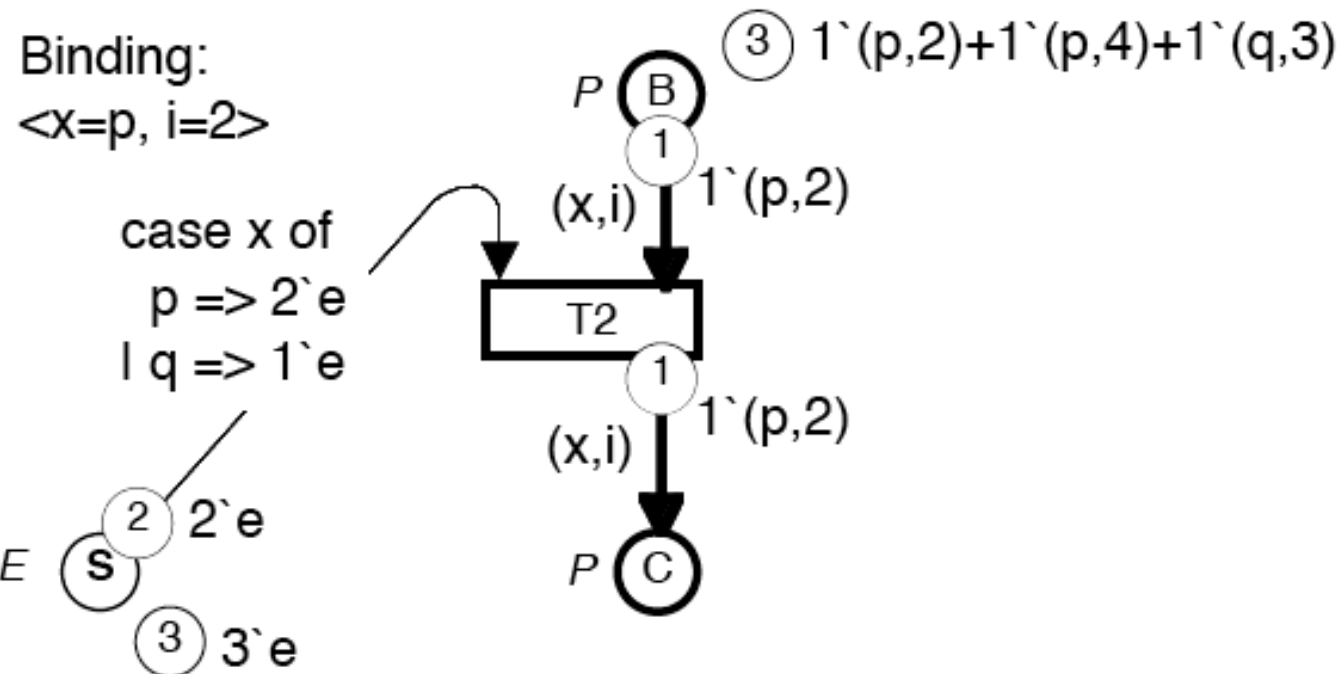
Strong type



Outro exemplo



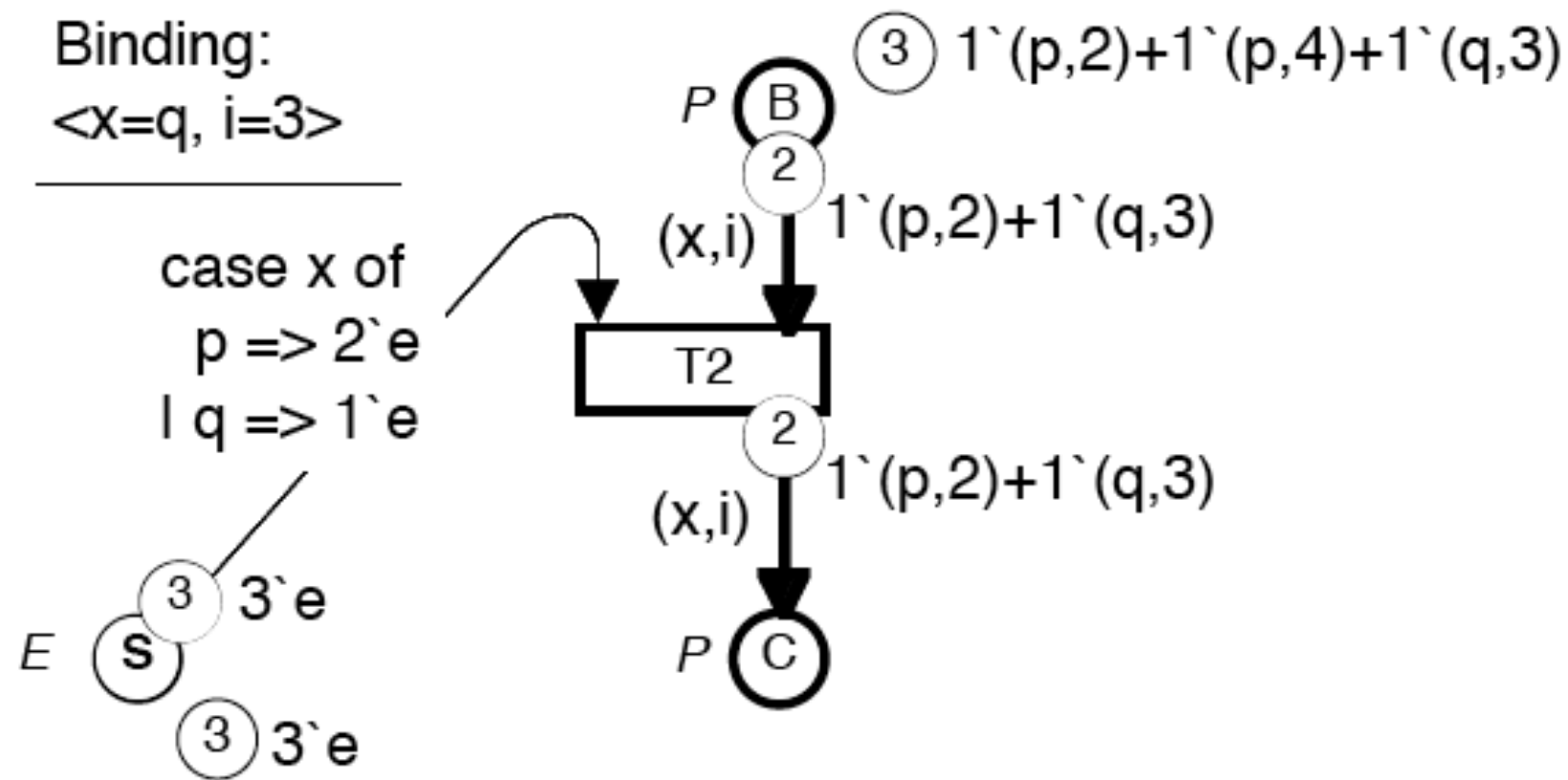
Uma vez respeitado o tipo qualquer possibilidade pode ocorrer.

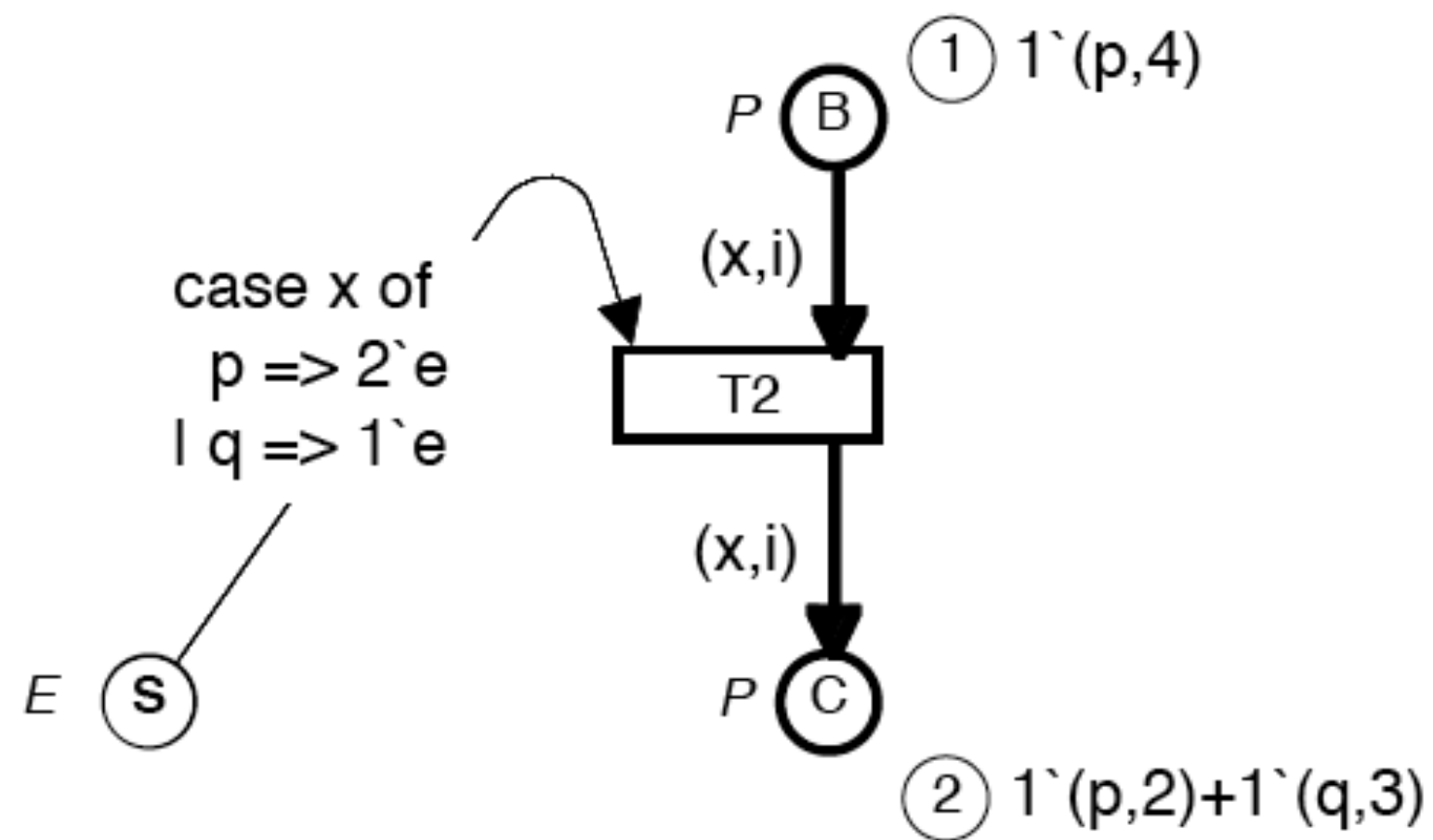


Concorrência

Binding:
 $\langle x=p, i=2 \rangle$

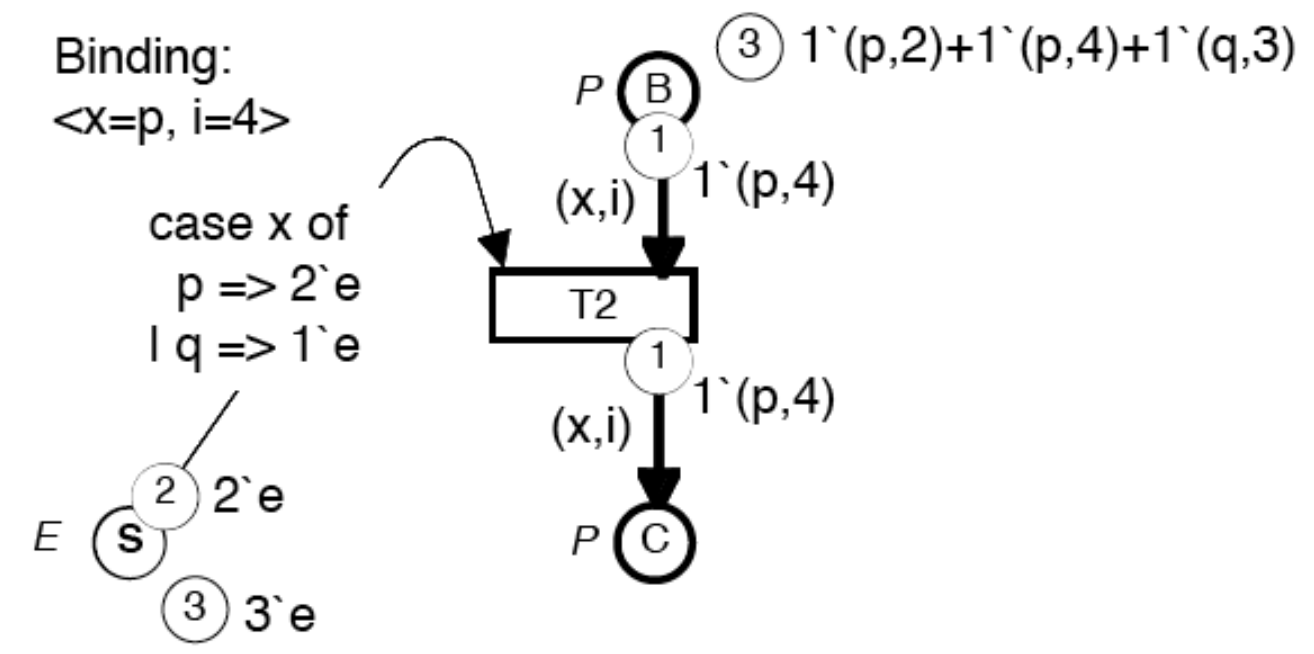
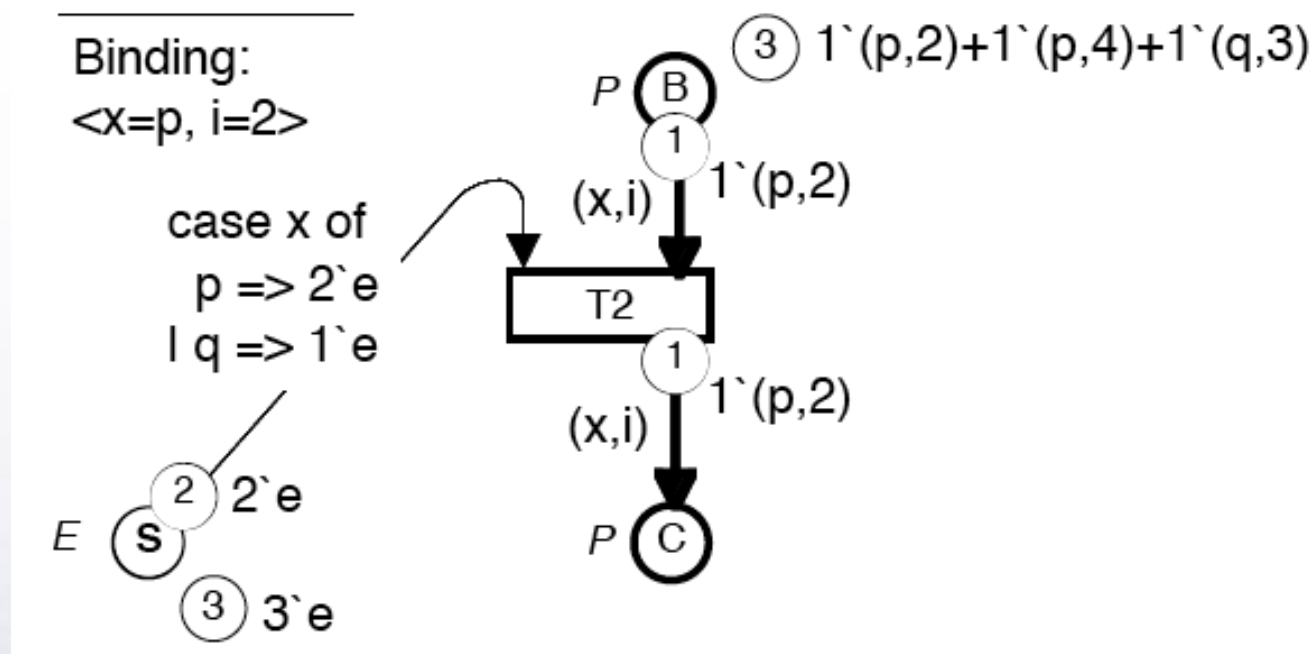
Binding:
 $\langle x=q, i=3 \rangle$





Gerenciando conflito

A situação de conflito é similar, só que agora levando em conta os tipos



CPN Elements

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 Condition

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: $(T2, \langle x=p, i=2 \rangle)$.

Biding

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

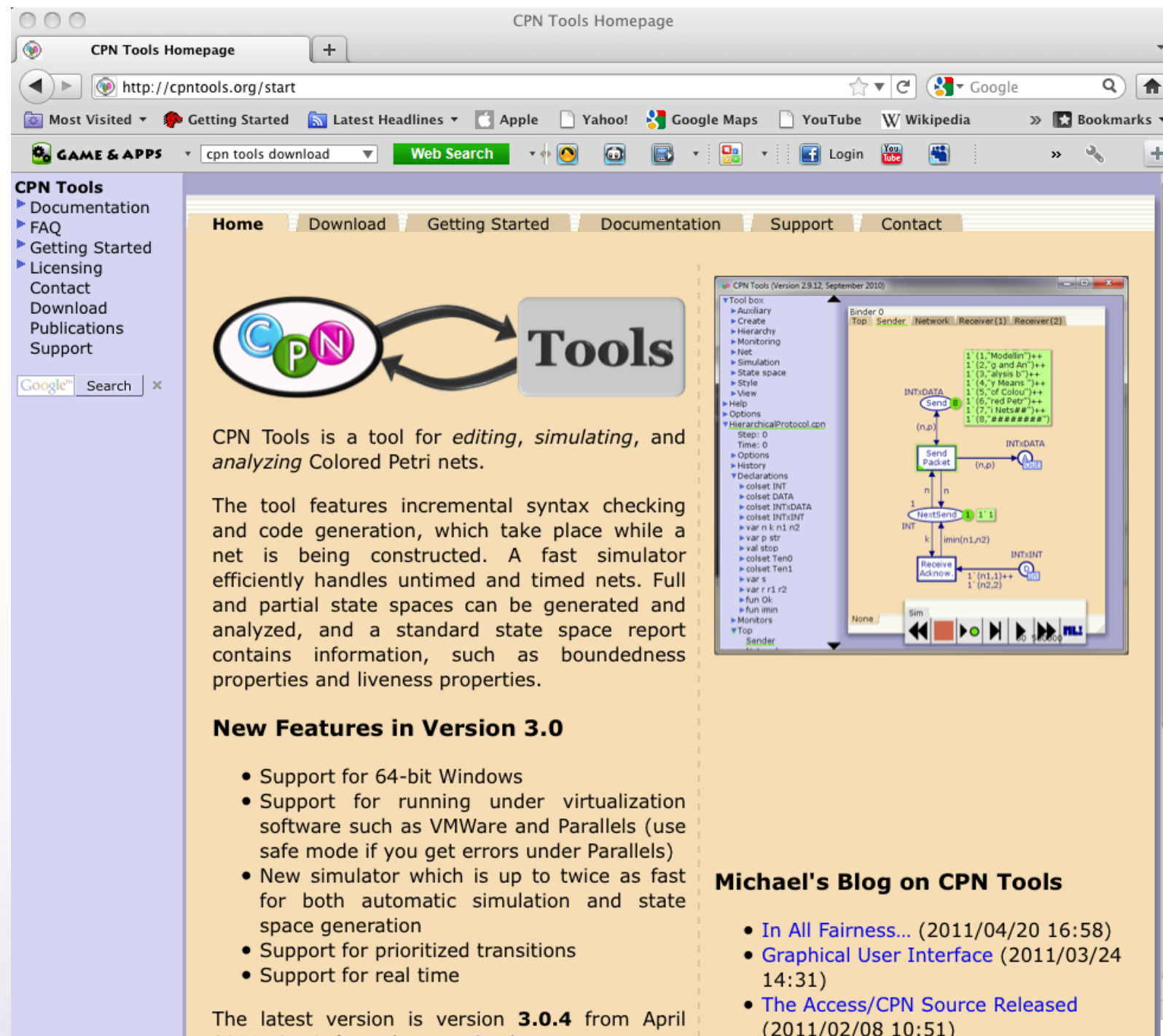
- There are *enough tokens* (of the correct colours on each input-place).
- The *guard* evaluates to true.

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".

CPN Tools (cpntools.org)



The screenshot shows the CPN Tools Homepage in a web browser. The browser's address bar displays <http://cpntools.org/start>. The page features a navigation menu with links to Home, Download, Getting Started, Documentation, Support, and Contact. A sidebar on the left lists various CPN Tools resources, including Documentation, FAQ, Getting Started, Licensing, Contact, Download, Publications, and Support. The main content area includes the CPN Tools logo, a description of the tool, and a list of new features in version 3.0.

CPN Tools

CPN Tools is 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.

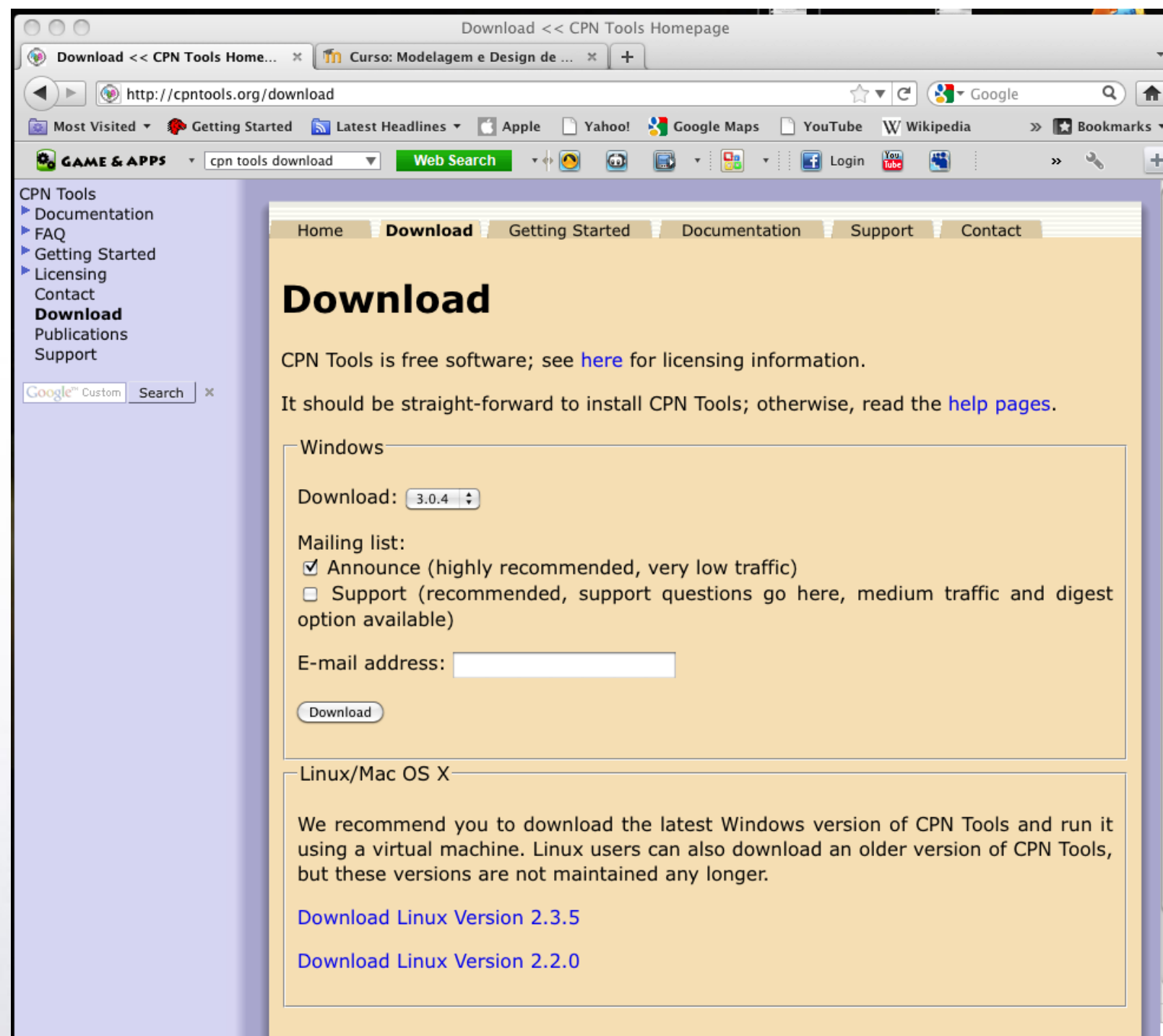
New Features in Version 3.0

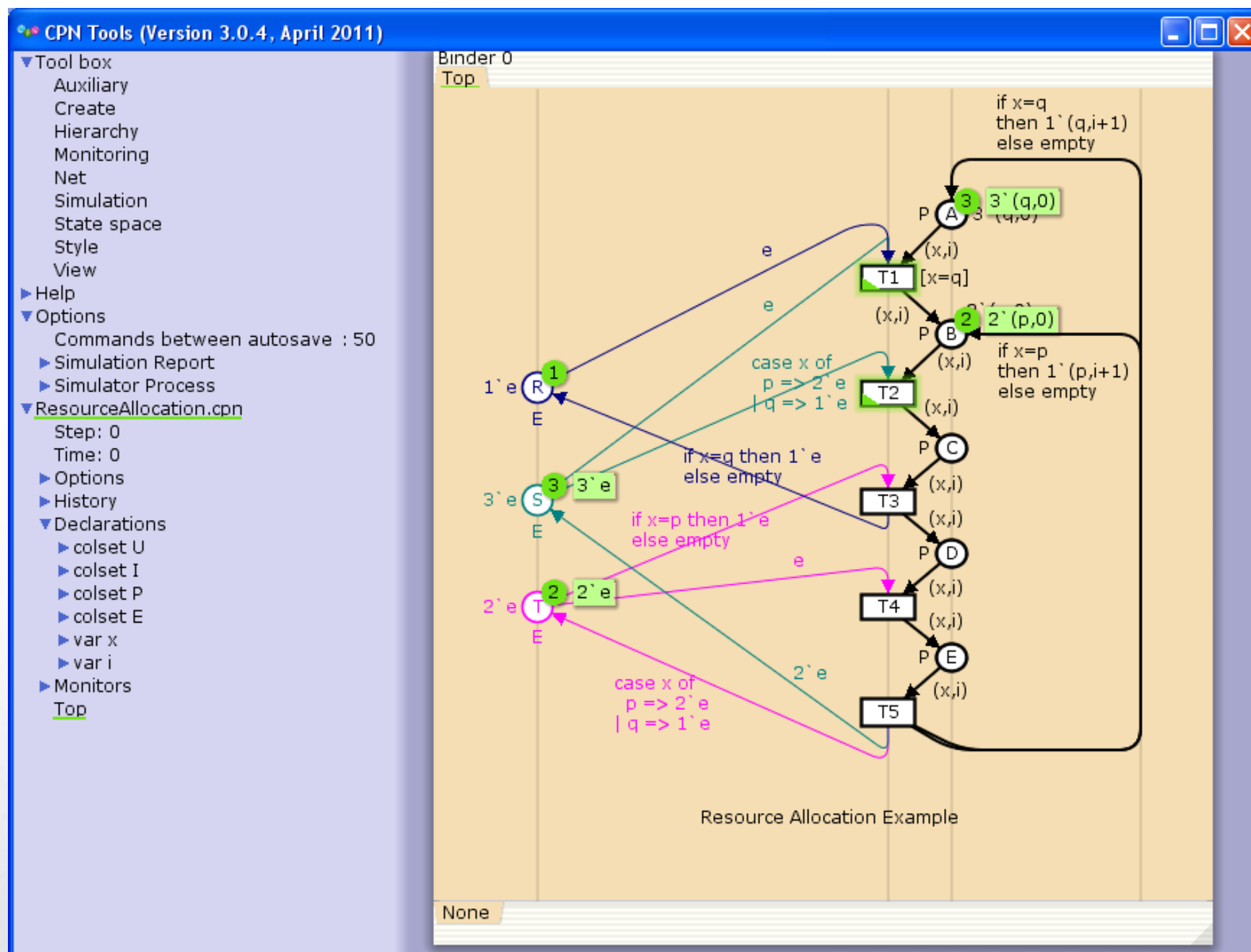
- Support for 64-bit Windows
- Support for running under virtualization software such as VMWare and Parallels (use safe mode if you get errors under Parallels)
- New simulator which is up to twice as fast for both automatic simulation and state space generation
- Support for prioritized transitions
- Support for real time

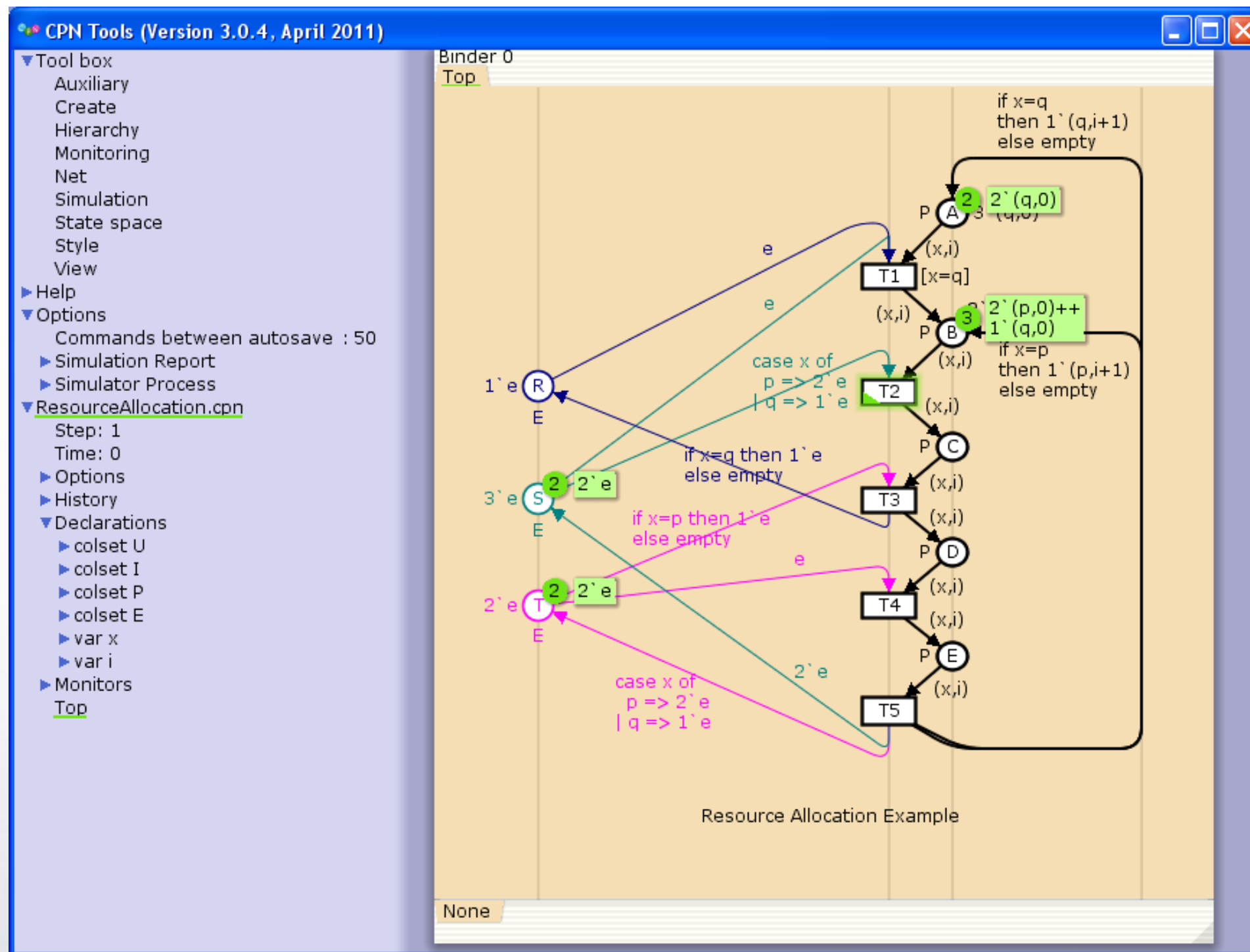
The latest version is version **3.0.4** from April

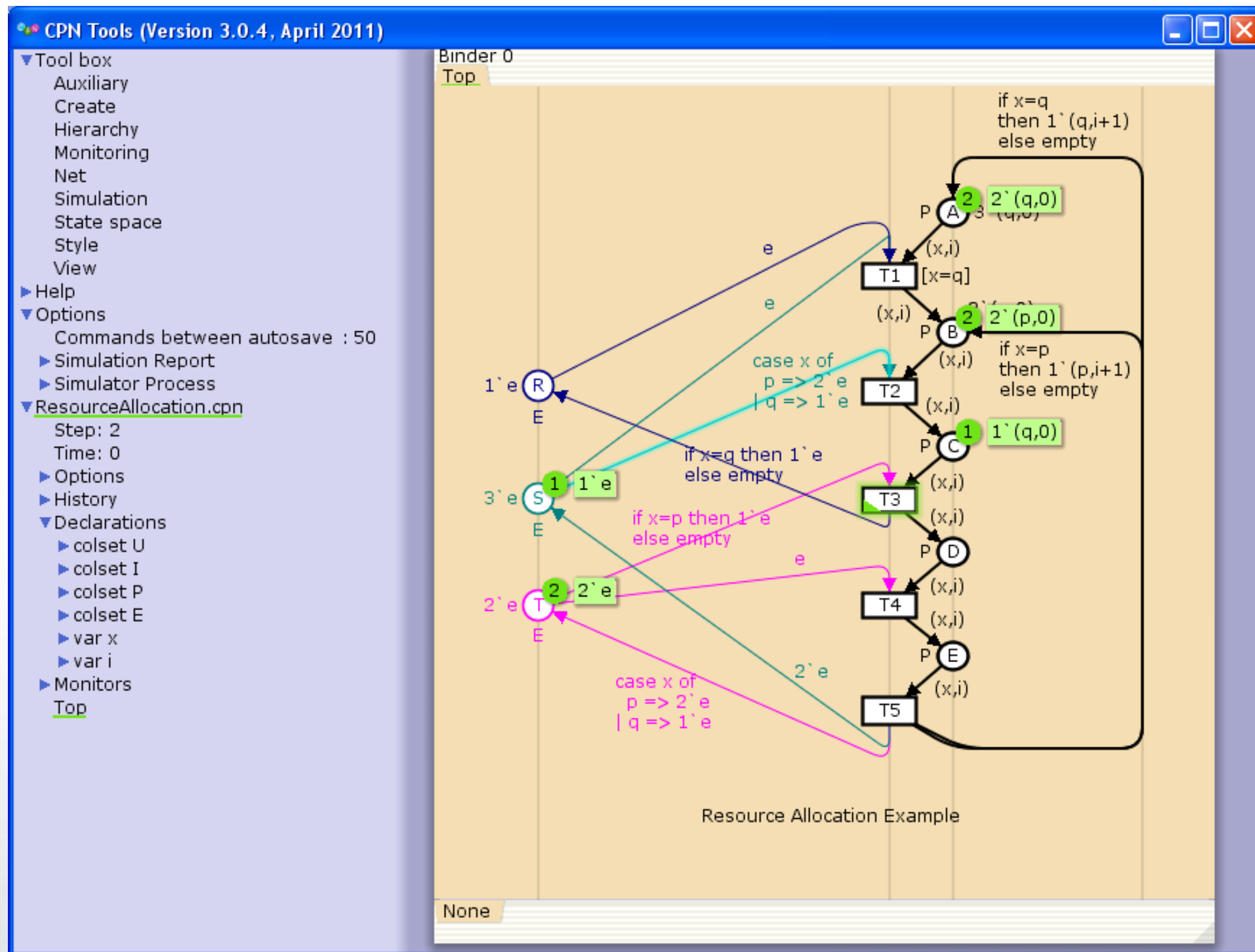
Michael's Blog on CPN Tools

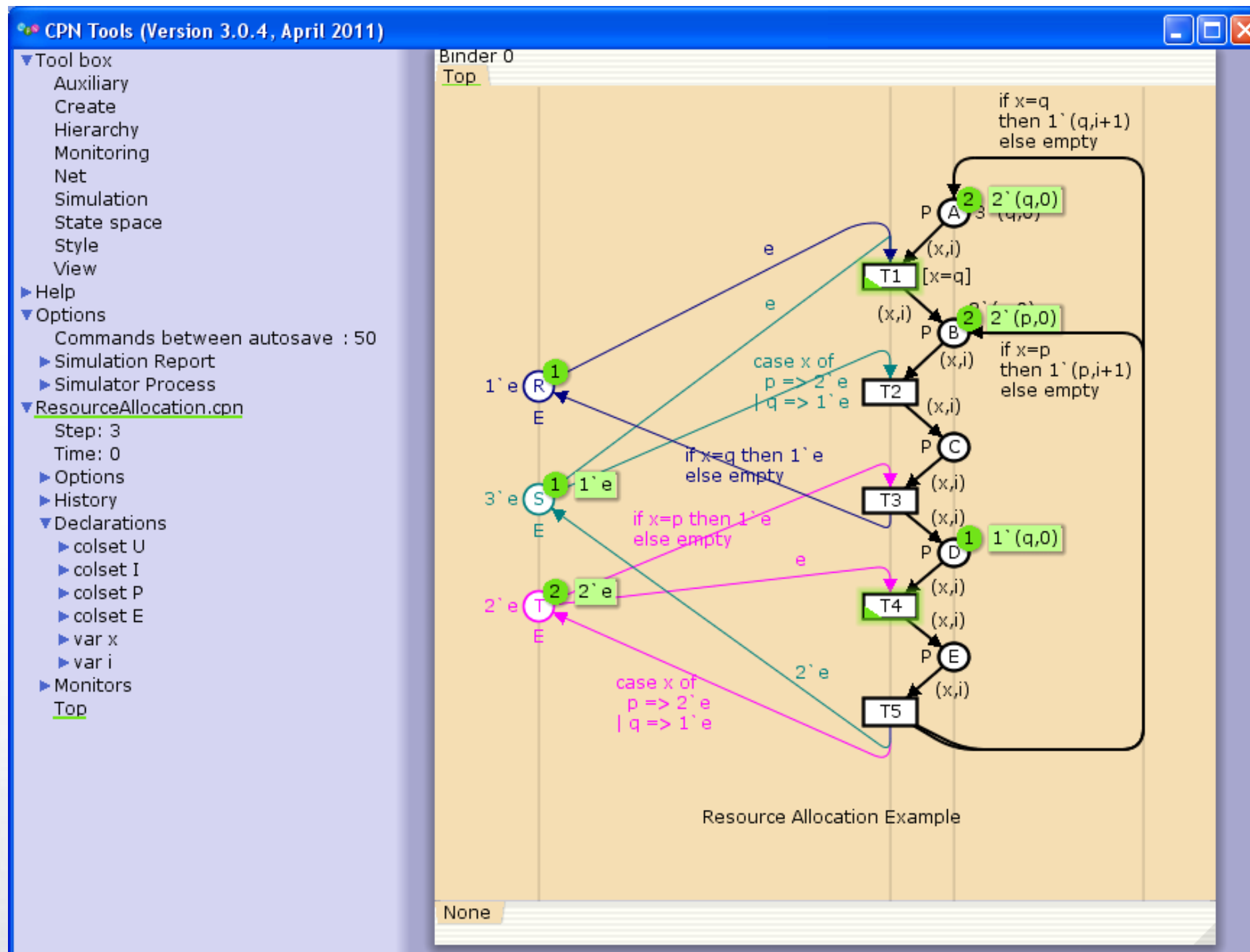
- [In All Fairness...](#) (2011/04/20 16:58)
- [Graphical User Interface](#) (2011/03/24 14:31)
- [The Access/CPN Source Released](#) (2011/02/08 10:51)

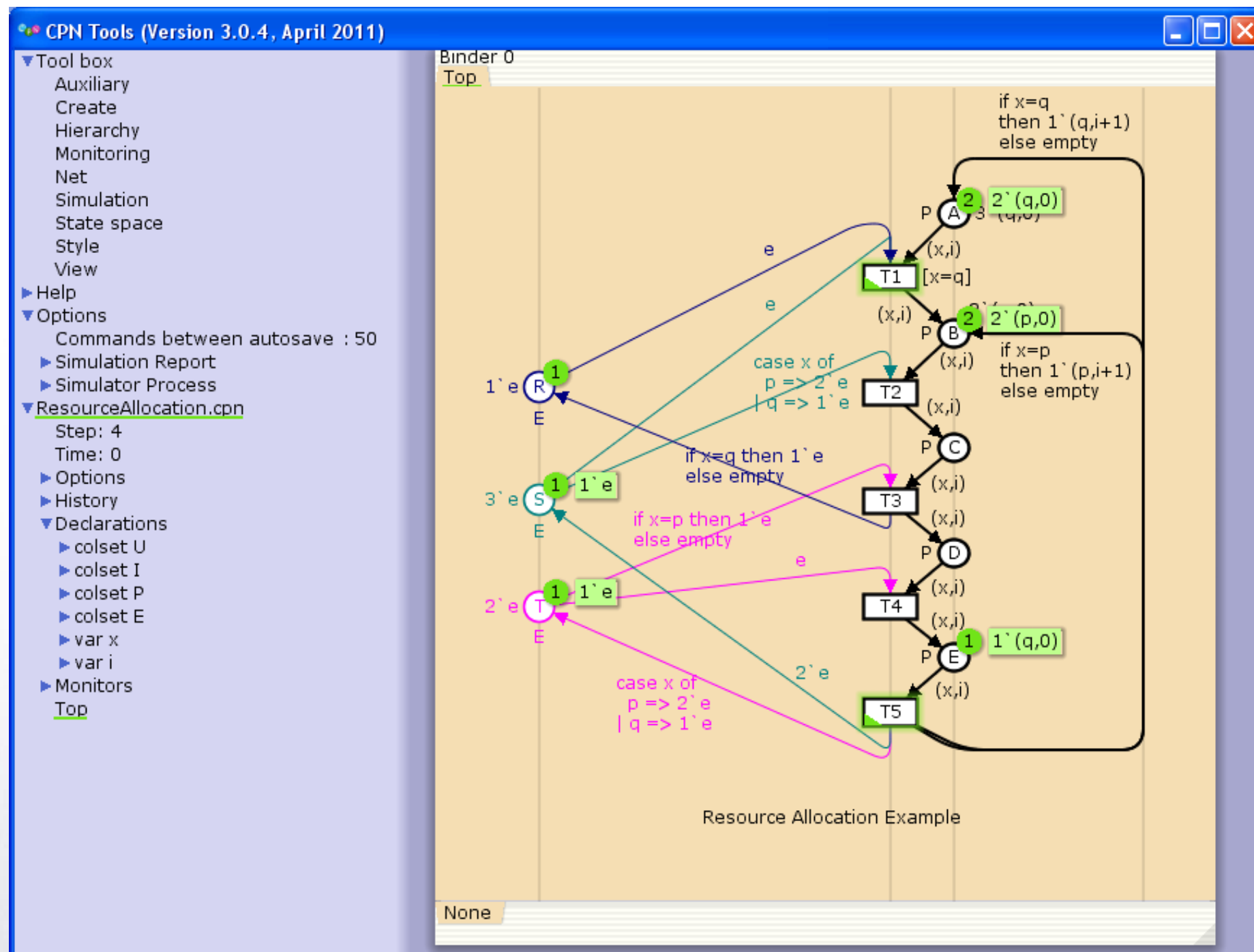


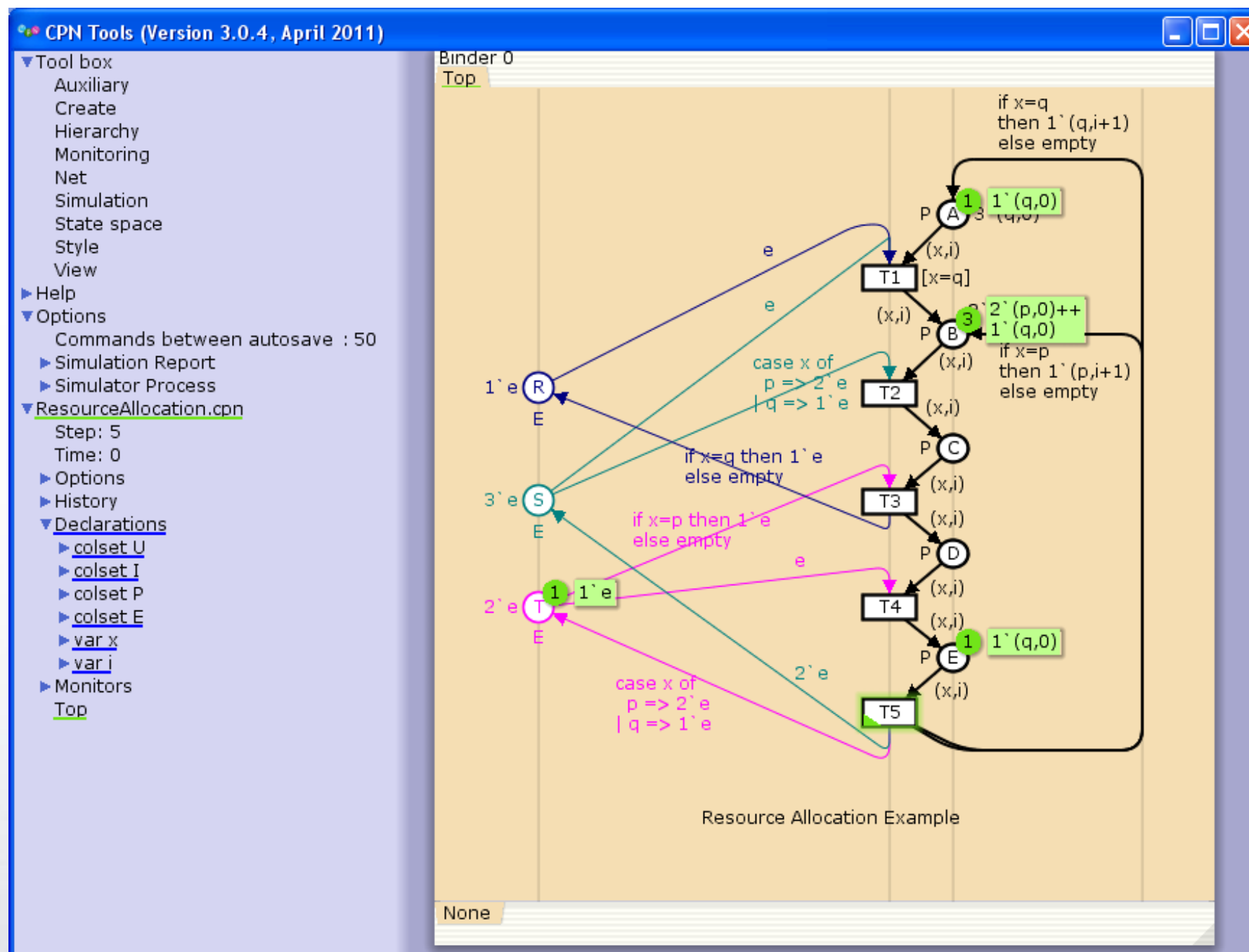


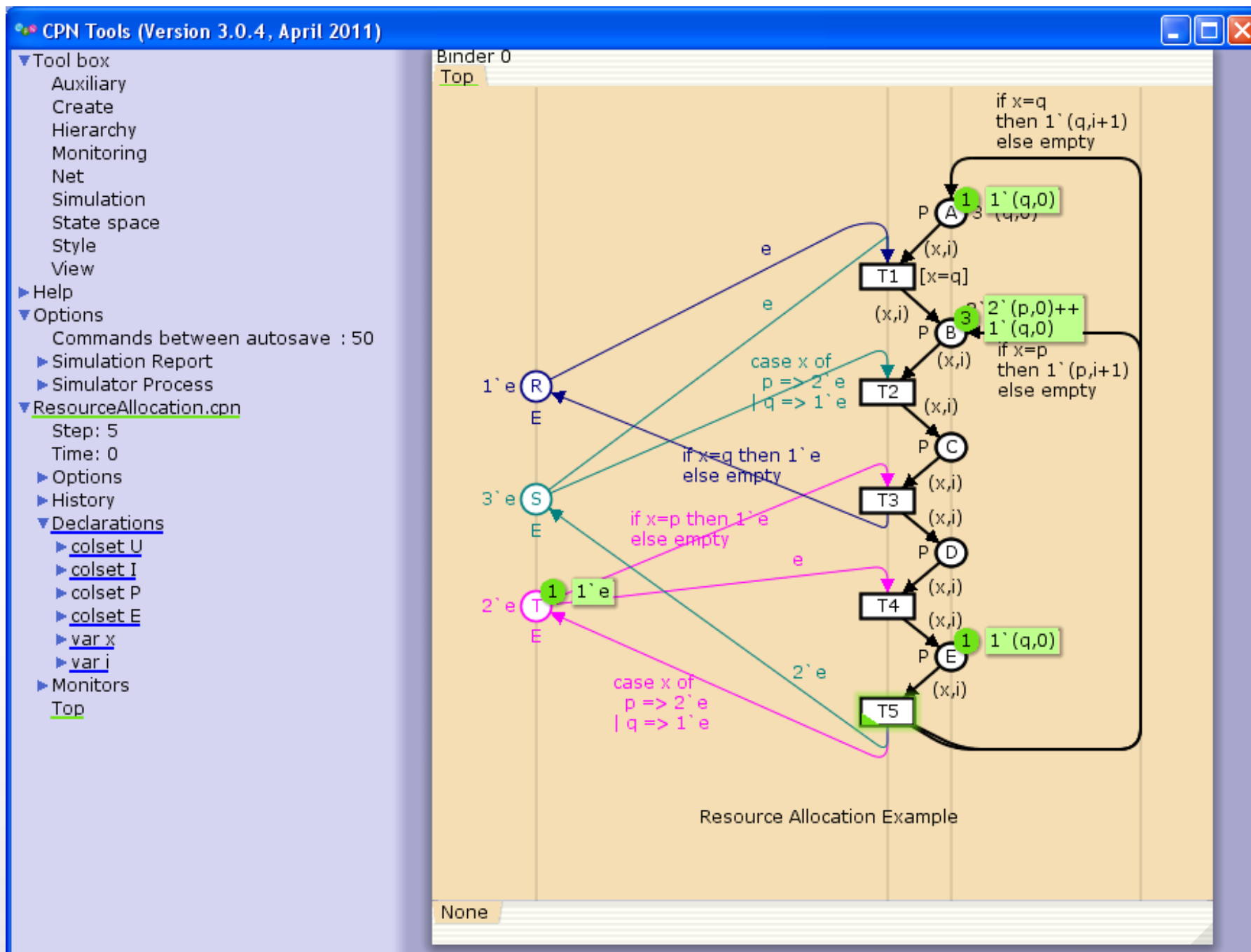


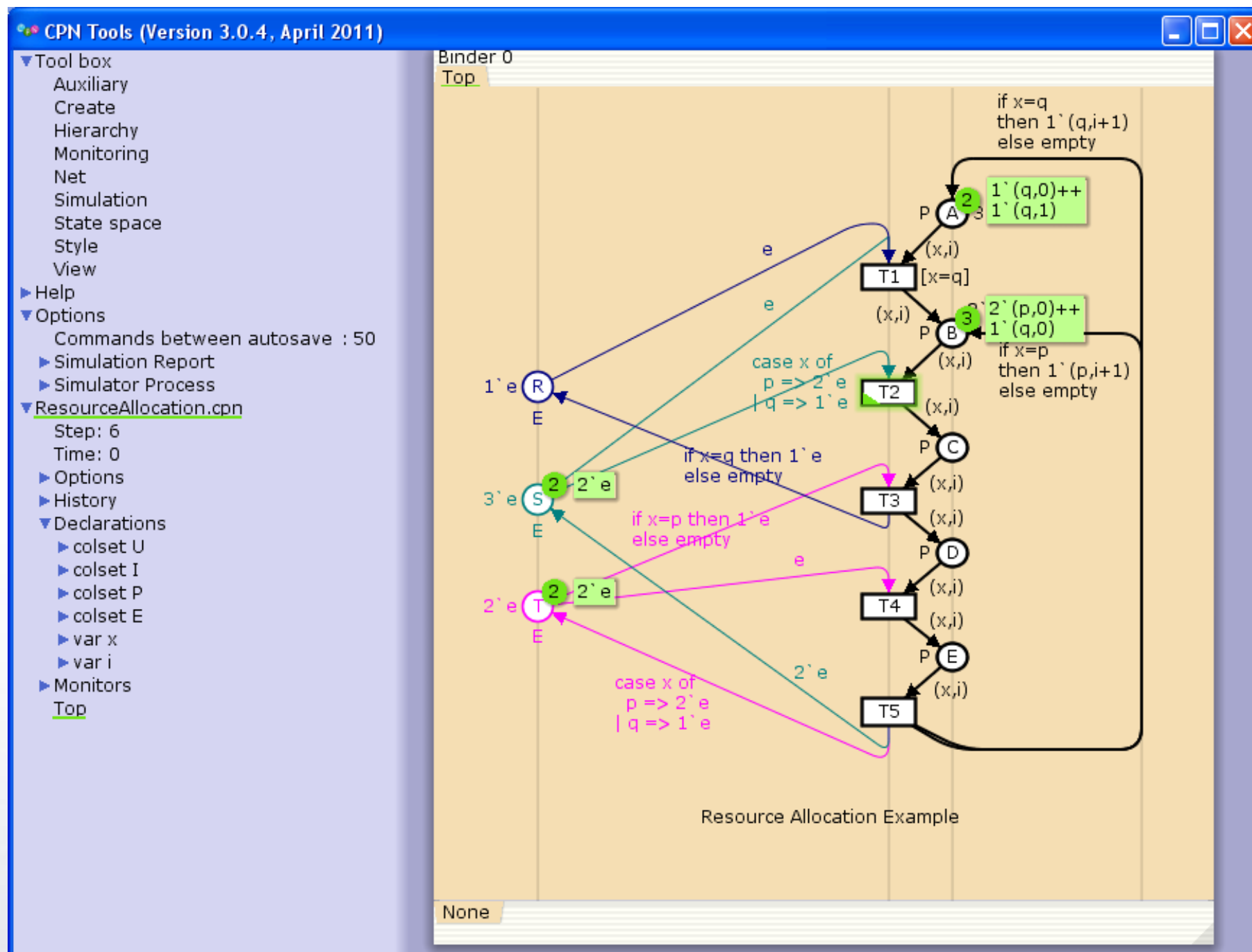








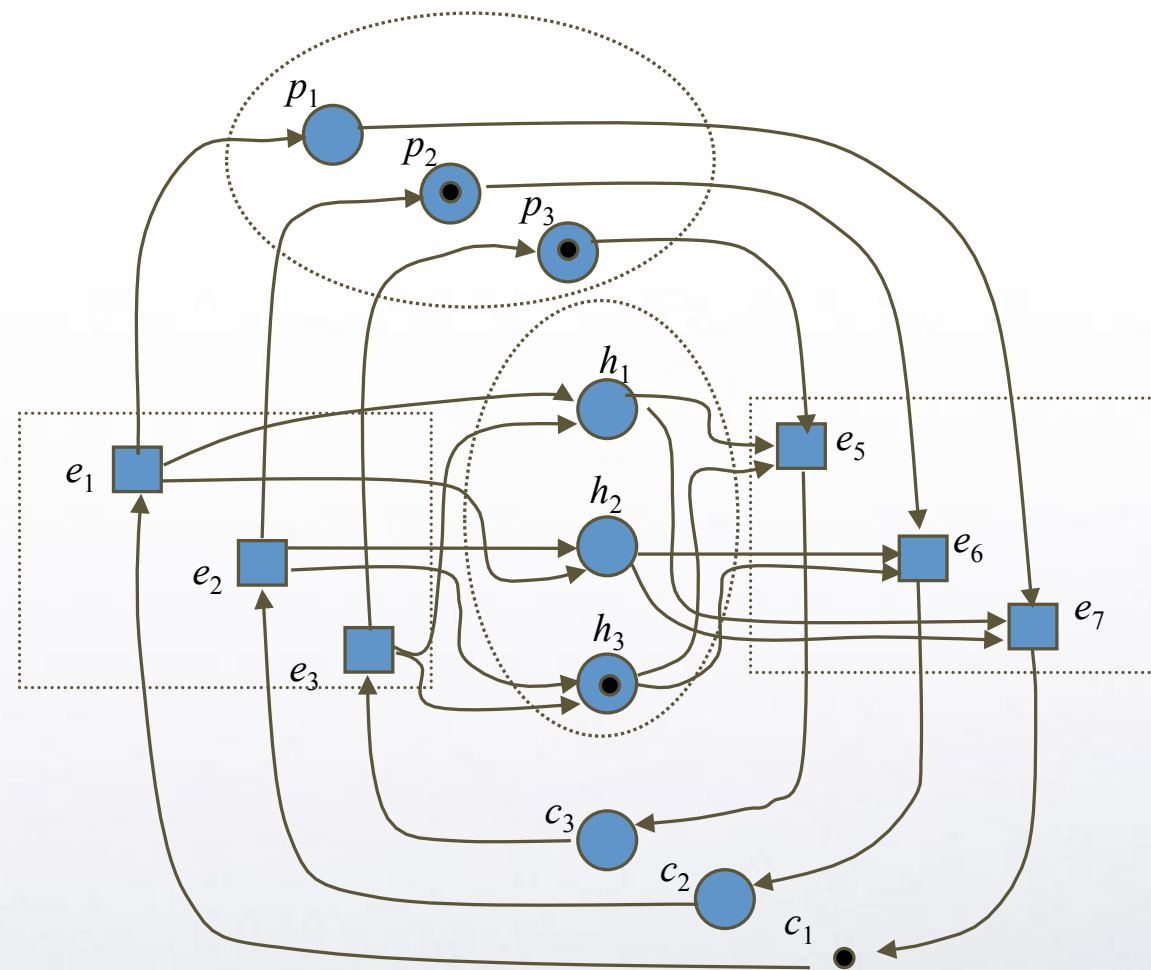
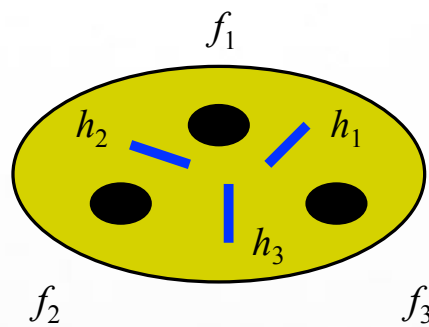


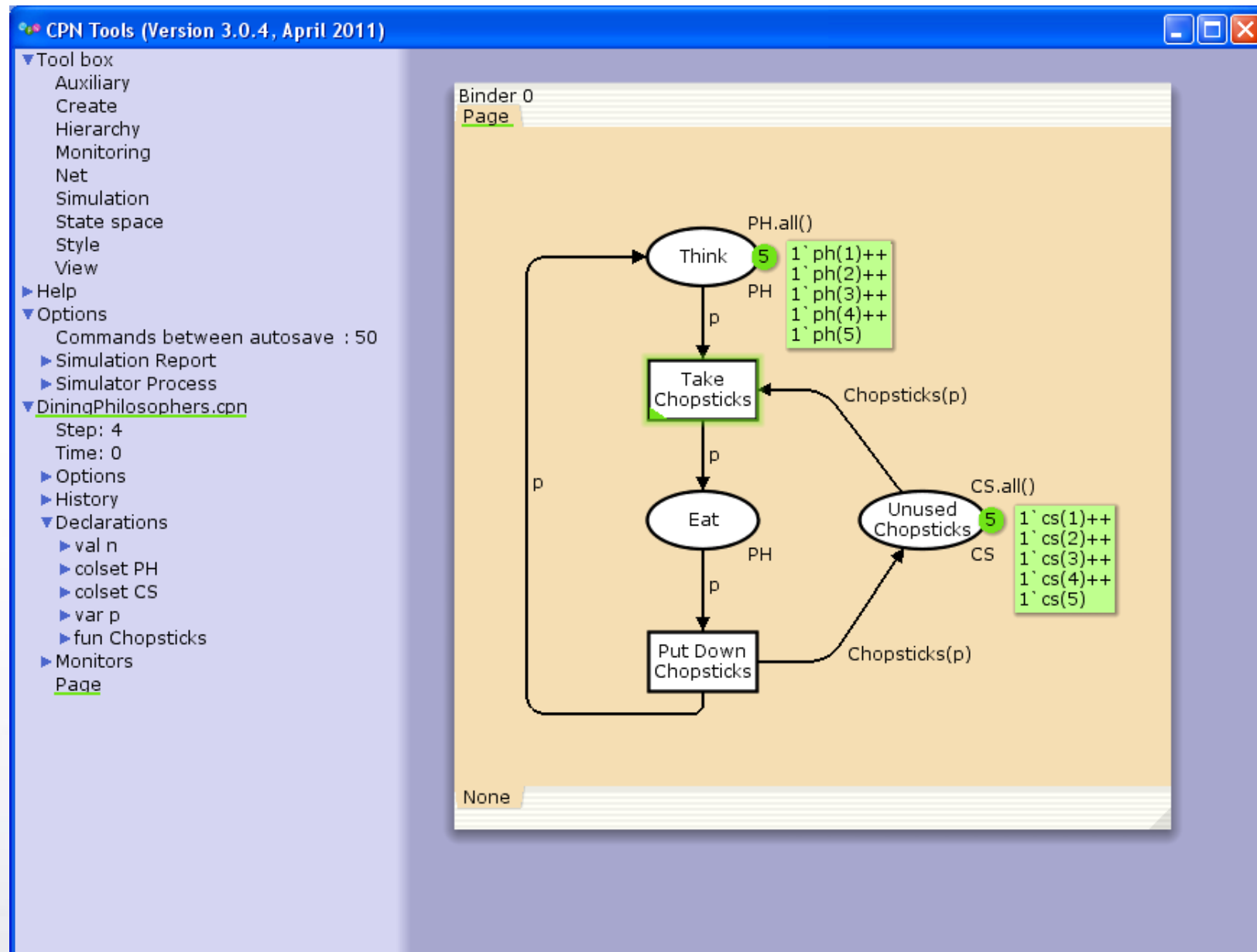




Modelagem clássica para o problema dos filósofos

O exemplo dos filósofos





Modelagem com CPNs

- Diferente abordagem, mais abstração
- Classificar elementos do projeto segundo tipos bem definidos
- A observância dos bindings (strong type) é fundamental
- Explorando as simetrias (pode não aparecer desta forma)
- Maior facilidade para reutilização

Alguns problemas...

- Diferente abordagem, mais abstração
Depende do background do designer, mas a questão base é escolher entre fazer a modelagem já direto em rede colorida ou começar pela rede clássica
- Classificar elementos do projeto segundo tipos bem definidos
Consequencia do item anterior e novamente é uma questão de disciplina.
- A observância dos bindings (strong type) é fundamental
Comum nos processos estruturados, choca-se com a excessiva carga funcional que os engenheiros costumam dar aos projetos.
- Explorando as simetrias (pode não aparecer desta forma)
De fato deve ser feito também no caso clássico, o tipo de simetria a ser explorada é que é diferente.
- Maior facilidade para reutilização
Será aplicado no tanto no caso clássico com no das redes de alto nível com a introdução da hierarquia, mas continua mais forte no caso das redes de alto nível, embora a escolha da componente reutilizável seja mais complexa.

Sobre o formalismo

Portanto, a única maneira de minorar os problema é voltar ao formalismo para explorar melhor a análise de propriedades, especialmente os invariantes.

O formalismo também abre a possibilidade de se ter outras interfaces mais amigáveis e adaptadas aos ambientes de trabalho, tendo o CPN Tools (por exemplo) por trás. Nesse caso a aderência ao formalismo das CPNs é importante e fundamental para a credibilidade da ferramenta.

CPN : definição formal

Definition: A Coloured Petri Net is a tuple $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ 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:
- $\forall t \in T: [\text{Type}(G(t)) = \text{Bool} \wedge \text{Type}(\text{Var}(G(t))) \subseteq \Sigma].$
- (viii) E is an **arc expression** function. It is defined from A into expressions such that:
- $\forall a \in A: [\text{Type}(E(a)) = C(p(a))_{MS} \wedge \text{Type}(\text{Var}(E(a))) \subseteq \Sigma]$ 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:
- $\forall p \in P: [\text{Type}(I(p)) = C(p)_{MS}].$

Fim