

Especificação, Modelação e Projecto de Sistemas Embutidos

Petri Nets

Paulo Pedreiras

pbrp@ua.pt



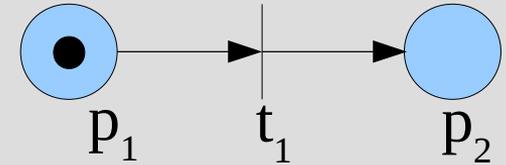
Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

Parcialmente baseado no livro “Modeling Embedded Systems and SOCs: concurrency and time in models of computation”
Axel Jantsch, Morgan Kaufmann Series in “Systems on Silicon”

V1.1 Novembro/2009

Redes de Petri: introdução

- Introduzidas em 1962 por Carl Adam Petri (tese de PhD)
- Foco na **modelação de concorrência**
- Foco no **acto de comunicar** e não nos dados comunicados
- **Comunicação** modelada por um **token** que **não contem dados**
- Todos os comportamentos que **não contribuem** para a **emissão** ou **consumo** de **tokens** são **omitidos** do modelo
- A **concorrência** pode ser estudada na sua **forma** mais **pura**

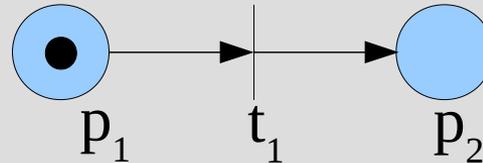


Petri Nets: Definição

Def.: Uma rede de Petri N é definida como $N=(P,T,A,w,\underline{x}_0)$, em que:

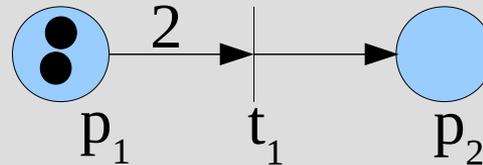
- **P** – Conjunto finito de lugares (*places*)
- **T** – Conjunto finito de transições (*transitions*)
- **A** – Conjunto de arcos/ramos (*arcs*)
 - $A \subseteq (P \times T) \cup (T \times P)$
- **w** - função de peso (*weight function*)
 - $w: A \rightarrow \mathbb{N}$
- **\underline{x}_0** – vector de marcação inicial (*initial marking vector*), equivalente à noção de estado inicial em máquinas de estados
 - $\underline{x}_0 \in \mathbb{N}^{|P|}$

Petri Nets: Exemplo



- Rede de Petri com dois lugares $\{p_1, p_2\}$ e uma transição $\{t_1\}$
 - As **tokens movem-se** de um lugar para outro por meio das **transições**
 - Cada transição “**consume**” w *tokens* quando dispara
 - Valor indicado por um número junto ao arco (1 se omitido)
 - A **marcação \underline{x}** indica quantos *tokens* estão em cada lugar
 - No exemplo temos $\underline{x}=[x(p_1), x(p_2)]=[1, 0]$

Petri Nets: transição permitida



Def. Seja a rede de Petri $N=(P,T,A,w,\underline{x}_0)$.

- $I(t)=\{p \in P \mid (p,t) \in A\}$ é o conjunto de **lugares de entrada** da transição t
- $O(t)=\{p \in P \mid (t,p) \in A\}$ é o conjunto de **lugares de saída** da transição t

Uma transição “**t**” está no estado **permitido** (*enabled*) se
 $x(p) \geq w(p,t) \forall p \in I(t)$

- De uma forma informal diz-se que **transição é permitida** se cada um dos seus **lugares de entrada** tem **peelo menos tantos tokens** quantos o **peso** do respectivo **arco**.
 - **No exemplo da figura a transição t_1 é permitida**

Petri Nets: função de transição

Def. Seja a rede de Petri $N=(P,T,A,w,\underline{x}_0)$, com $P=\{p_0, \dots, p_{n-1}\}$ lugares e $\underline{x}=[x(p_0), \dots, x(p_{n-1})]$ a respectiva marcação.

- A **função de transição** $G:(\mathbb{N}^{|\mathcal{I}|} \times T) \rightarrow \mathbb{N}^{|\mathcal{I}|}$ define-se como

$$G(\underline{x}, t) = \begin{cases} \underline{x}' & \text{se } x(p) \geq w(p, t) \quad \forall p \in I(t) \\ \underline{x} & \text{caso contrário} \end{cases}$$

com $x'(p_i) = x(p_i) - w(p_i, t) + w(t, p_i)$, para $0 \leq i \leq n$

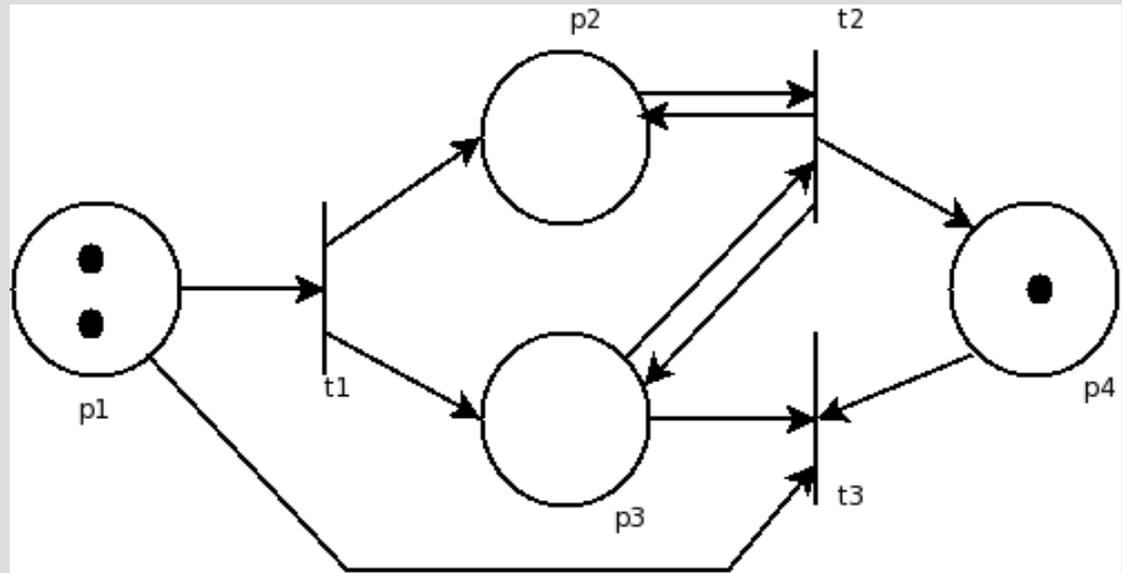
Informalmente:

- Se uma transição não é possível a marcação não se altera
- Quando uma **transição** t (permitida) é “**disparada**”:
 - i. O número de *tokens* em cada $p \in I(t)$ é **reduzida** de $w(p, t)$
 - ii. O número de *tokens* em cada $p \in O(t)$ é **aumentada** de $w(t, p)$

Petri Nets: função de transição

- **Eg.** Considere a rede de Petri $N=(P,T,A,w,\underline{x}_0)$, com:

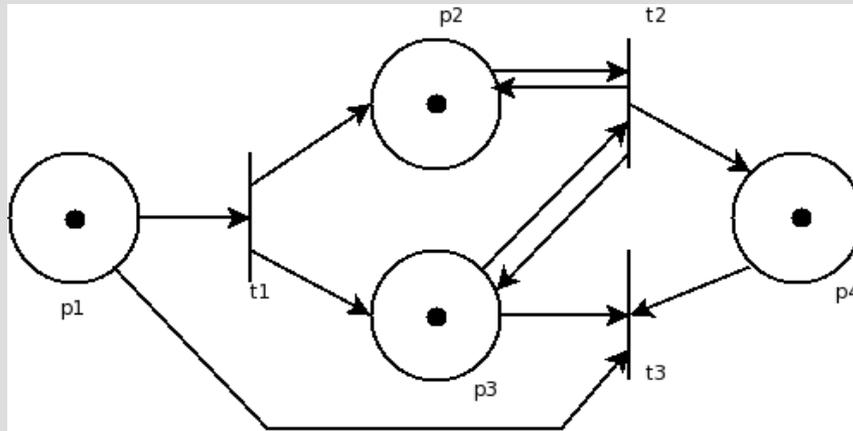
- $P=\{p_1, \dots, p_4\}$, $T=\{t_1, t_2, t_3\}$,
- $A=\{(p_1, t_1), (p_1, t_3), (p_2, t_2), (p_3, t_2), (p_3, t_3), (p_4, t_3), (t_1, p_2), (t_1, p_3), (t_2, p_2), (t_2, p_3), (t_2, p_4)\}$,
- $w(a)=1 \forall a \in A$,
- $x_0=[2, 0, 0, 1]$



- Quais as transições permitidas?

Petri Nets: função de transição

- E.g. (cont): ... Após o disparo de t_1



$$X_0 = [2, 0, 0, 1]$$

↓

$$X_1 = [1, 1, 1, 1]$$

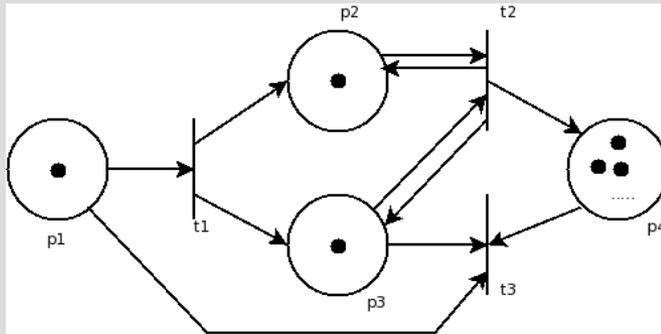
Agora t_1, t_2 e t_3 são permitidas! Qual(is) dela(s) deve(m) disparar?

A semântica das Petri Nets não o define!
Assim, **as PN são inerentemente não determinísticas!**

Petri Nets: função de transição

- E.g. (cont): Duas possibilidades:

i) **t2 pode disparar arbitrariamente**, acumulando tokens em p4, ou



$$\underline{x}_0 = [2, 0, 0, 1]$$

↓ (t1)

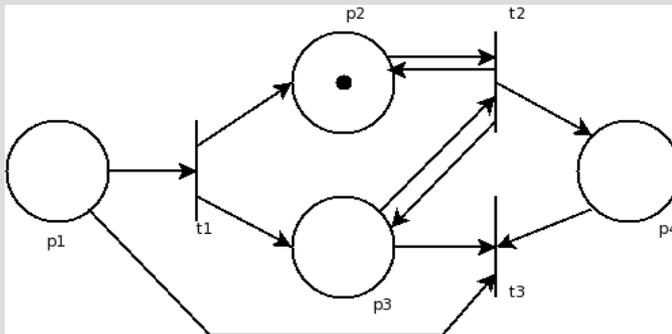
$$\underline{x}_1 = [1, 1, 1, 1]$$

... (t2)

$$\underline{x}_n = [1, 1, 1, n]$$

... (t2)

ii) **t3 dispara**, levando ao estado $\underline{x}_2 = [0, 1, 0, 0]$ em que não há transições permitidas



$$\underline{x}_0 = [2, 0, 0, 1]$$

↓ (t1)

$$\underline{x}_1 = [1, 1, 1, 1]$$

↓ (t3)

$$\underline{x}_2 = [0, 1, 0, 0]$$

Petri Nets: Conjunto de Alcançabilidade (*Reachability set*)

Def. “Alcançabilidade” (*Reachability*)

- Para a PN $N=(P,T,A,w,\underline{x}_0)$ e um dado estado \underline{x} , um estado \underline{y} diz-se **imediatamente alcançável** a partir de \underline{x} se existe uma transição $t \in T$ tal que $G(\underline{x},t)=\underline{y}$.
- O **conjunto de alcançabilidade** $R(\underline{x})$ é o menor conjunto de estados definidos por:
 - i. $\underline{x} \in R(\underline{x})$
 - ii. Se $\underline{y} \in R(\underline{x})$ e $\underline{z}=G(\underline{y},t)$ para algum $t \in T$, então $\underline{z} \in R(\underline{x})$.

Informalmente, o conjunto de alcançabilidade de um estado inclui **todos os estados** que **possam** eventualmente ser **atingidos disparando** repetidamente **transições**

Petri Nets: Matriz de incidência

Def. Seja a rede de Petri $N=(P,T,A,w,\underline{x}_0)$, com

$P=\{p_1, \dots, p_n\}$ e $T=\{t_1, \dots, t_m\}$.

- O **vector de disparo** $\underline{u}=[0, \dots, 0, 1, 0, \dots, 0]$ é o vector de dimensão m em que $1 \leq j \leq m$ corresponde à transição t_j . Todas as entradas do vector excepto uma são zero.
- A **matriz de incidência** A é uma matriz $m \times n$ cuja entrada (j,i) é $a_{j,i} = w(t_j, p_i) - w(p_i, t_j)$

A matriz de incidência contém informação acerca do efeito líquido de disparar a transição t_j no lugar p_i . A matriz de incidência permite **calcular a evolução de uma rede de Petri**, por meio da denominada **equação de estado**

$$\underline{x}' = \underline{x} + \underline{u} \cdot A$$

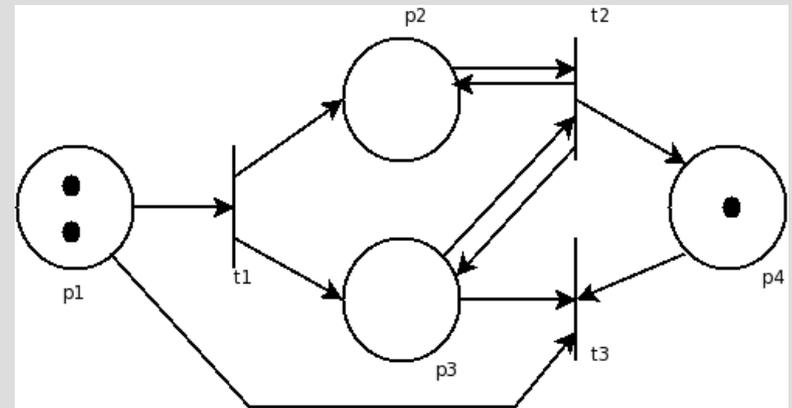
Petri Nets: Uso da matriz de incidência

- **E.g.** Considere o exemplo apresentado no slide 7

$$\mathbf{A} = \begin{pmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & -1 \end{pmatrix} \quad t_1 \text{ dispara}$$

$$\underline{x}_0 = [2, 0, 0, 1]$$

$$\underline{u}_1 = [1, 0, 0], \quad \underline{u}_2 = [0, 1, 0], \quad \underline{u}_3 = [0, 0, 1]$$



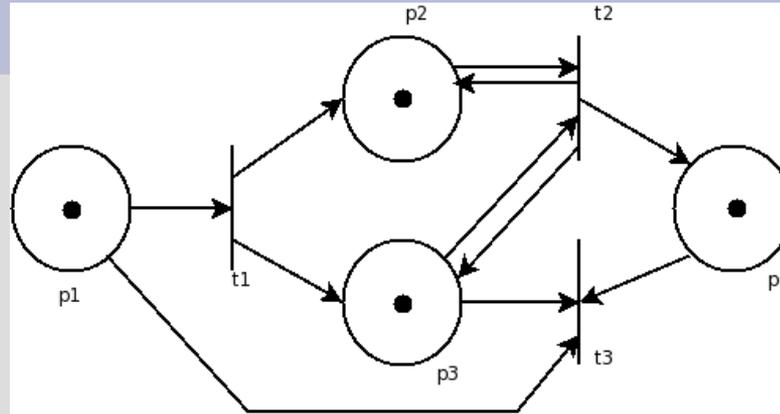
- **Passo 1.** t_1 dispara:

$$\begin{aligned} x_1 &= x_0 + u_1 \cdot \mathbf{A} = [2, 0, 0, 1] + [1, 0, 0] \cdot \begin{pmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & -1 \end{pmatrix} \\ &= [2, 0, 0, 1] + [-1, 1, 1, 0] = [1, 1, 1, 1] \end{aligned}$$

que é o mesmo resultado obtido por análise gráfica!

Petri Nets: Uso da matriz de incidência

E.g. (cont).



- Passo 2. t_3 dispara:

$$\underline{x}_2 = \underline{x}_1 + \underline{u}_2 \cdot \mathbf{A} = [1, 1, 1, 1] + [0, 0, 1]^* \begin{pmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & -1 \end{pmatrix}$$

$$= [1, 1, 1, 1] + [-1, 0, -1, -1] = [0, 1, 0, 0]$$

- Passo 3. t_3 dispara novamente:

$$\underline{x}_3 = \underline{x}_2 + \underline{u}_2 \cdot \mathbf{A} = [0, 1, 0, 0] + [0, 0, 1]^* \begin{pmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & -1 \end{pmatrix}$$

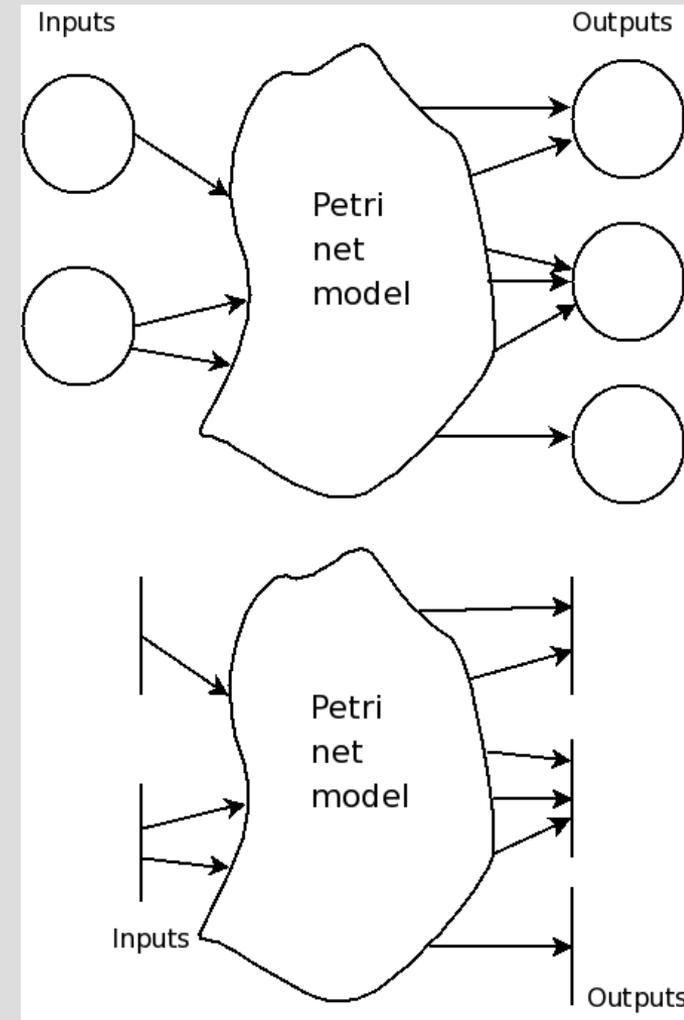
$$= [0, 1, 0, 0] + [-1, 0, -1, -1] = [-1, 1, -1, -1]$$

Vector de estado com valores negativos!! **Transições inválidas!**

Petri Nets: Modelação de entradas e saídas

Modelação explícita de I/O

- **Como lugares sem ramos** de entrada (inputs) e lugares sem ramos de saída (outputs)
 - Presume-se que o “ambiente” coloca *tokens* nos lugares de entrada e os consome dos lugares de saída
- **Como transições**
 - O ambiente controla quando e quais as transições de entrada que devem disparar, e consome os *tokens* emitidos pelas transições de saída



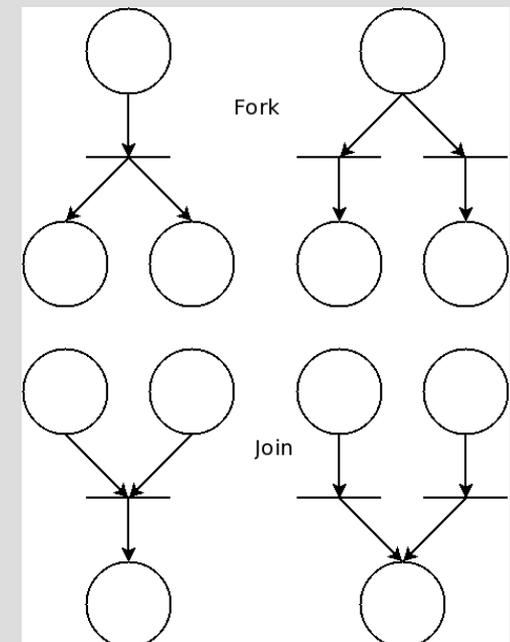
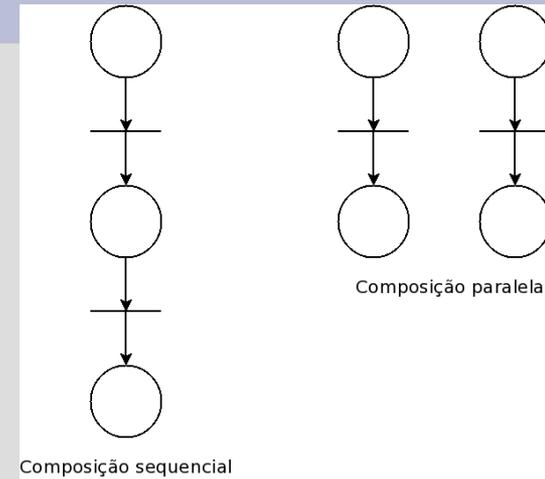
Templates para casos-padrão

Composição de duas redes

- A combinação de duas redes pode ser:
 - Composição **sequencial**:
 - Uma rede tem de produzir *tokens* consumidos pela outra rede
 - Composição em **paralelo**:
 - Transições podem ocorrer em paralelo (desde que haja *tokens* disponíveis)

Fork e Join

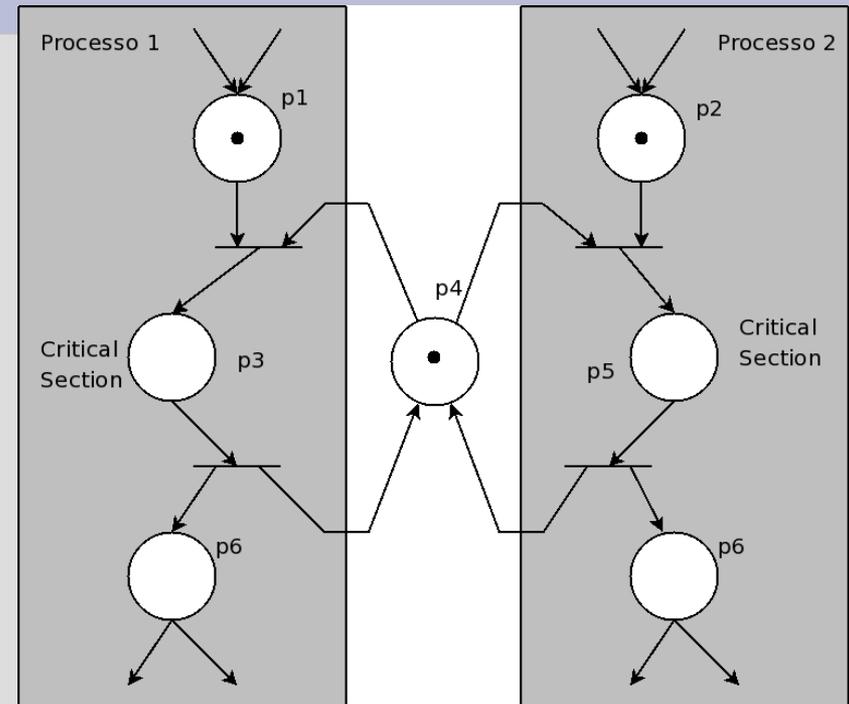
- Divisão e agrupamento de fluxos de controlo



Templates para casos-padrão

Exclusão mútua

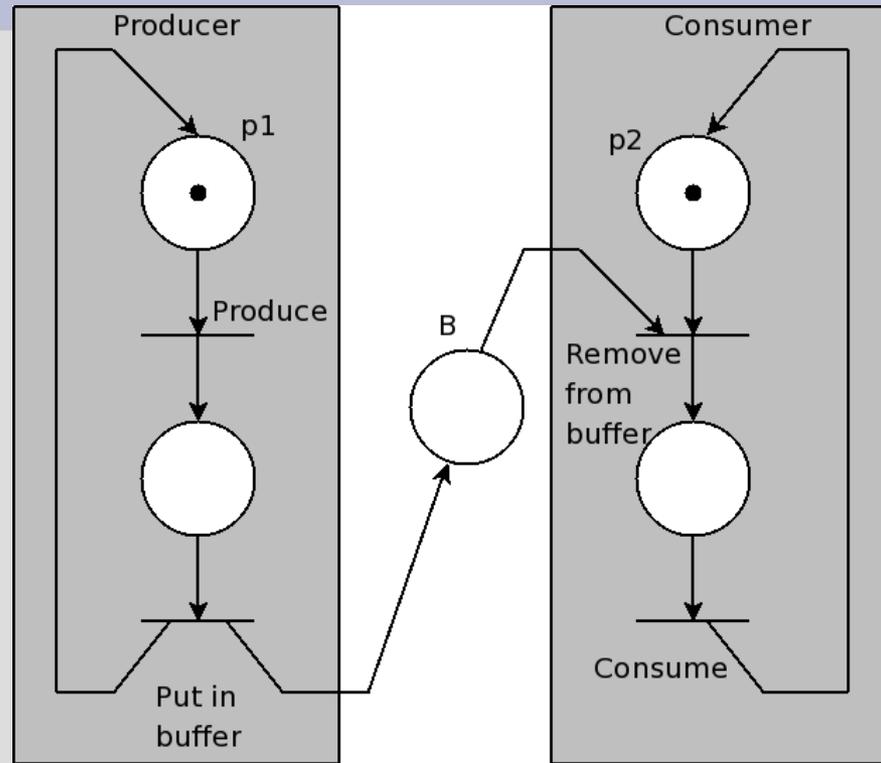
- Os dois processos **não** entram em **simultâneo** na região crítica;
- Facilmente generalizável a um **número arbitrário de processos**
- Permite também **limitar a k** o número de processos que podem **usar** um certo **recurso**
 - k é a marcação inicial de p_4



Templates para casos-padrão

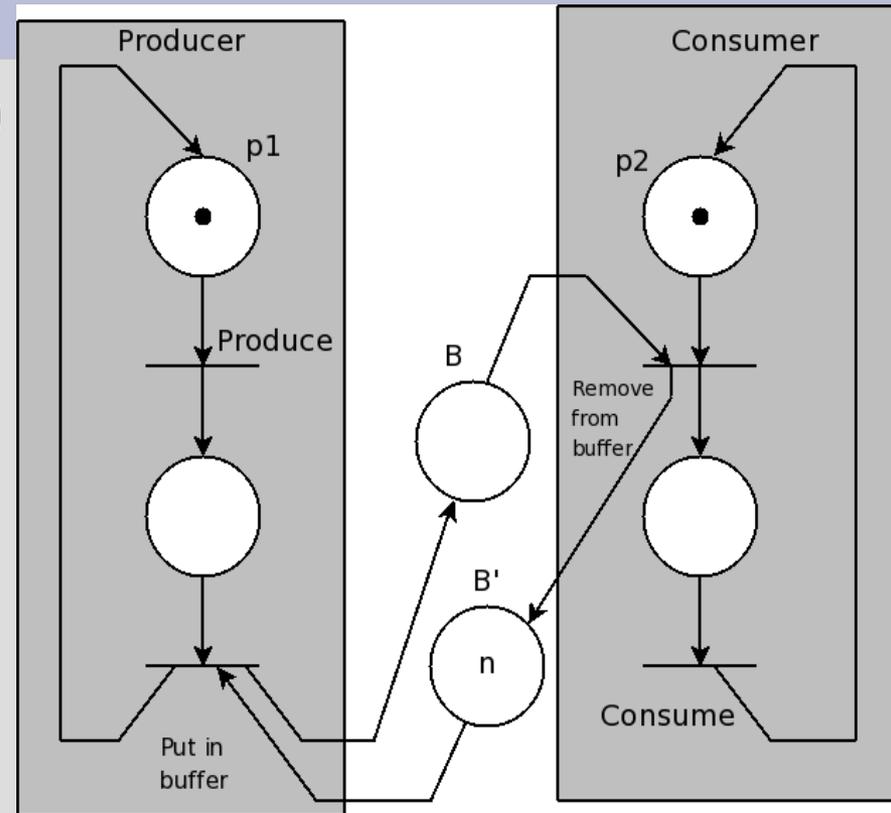
Produtor/Consumidor (1)

- O **produtor coloca** dados num **buffer** e o **consumidor** posteriormente **lê** os dados do mesmo **buffer**
- Calculando o número máximo de *tokens* que podem aparecer em B é possível **calcular** a **dimensão** máxima do **buffer** de dados



Templates para casos-padrão

Produtor/Consumidor (2) *buffer finito*



- Com uma pequena modificação é possível **condicionar** a **ativação** do **produtor** à **disponibilidade** de espaço no *buffer*
- Inicializa-se** o lugar B' com o **n tokens**, em que n corresponde à **dimensão** máxima do *buffer*

Métodos de análise para redes de Petri

A **modelação** de sistemas tem por **objectivo geral** responder a perguntas como:

- **Ir**á o sistema alguma vez entrar num certo **estado particular**?
- Será o sistema sempre capaz de **evitar** certos **estados “perigosos”** ?
- Será o sistema sempre capaz de **reagir** a *inputs*?
- Será que o sistema alguma vez **alcançará** um certo estado desejado?

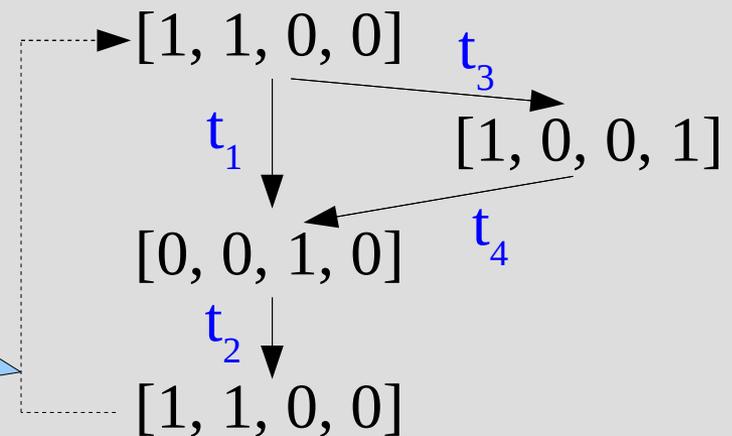
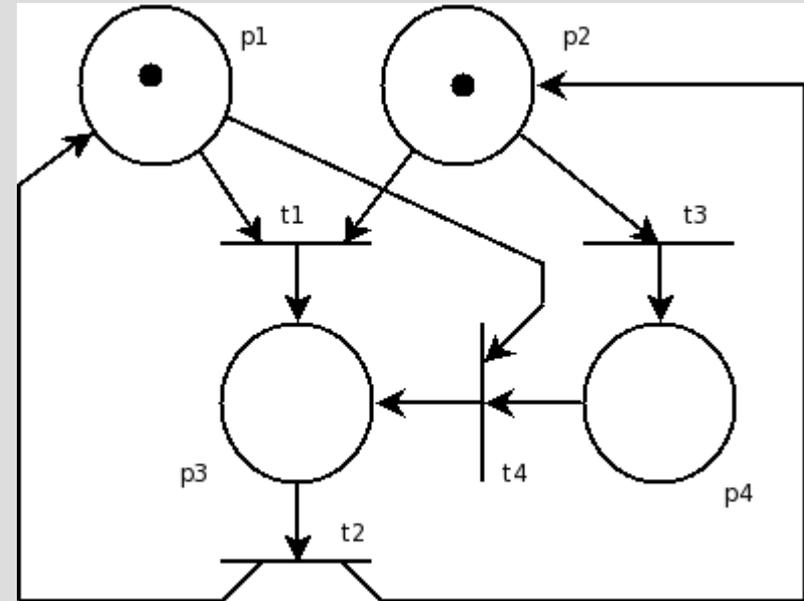
Nos slides seguintes iremos ver **se** e **como** é que estas perguntas podem ser respondidas no contexto das redes de Petri.

A árvore de cobertura (*coverability tree*)

- Informalmente define-se a **árvore de cobertura** como uma árvore em que os **arcos** representam **transições** e os **nós** denotam o conjunto de **estados** que podem ser atingidos por uma sequência de transições.

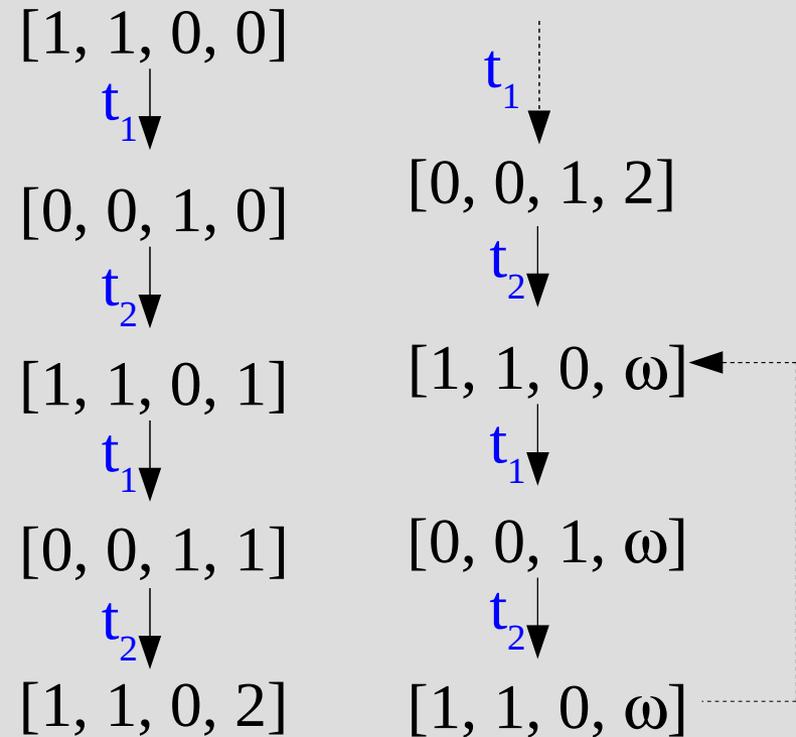
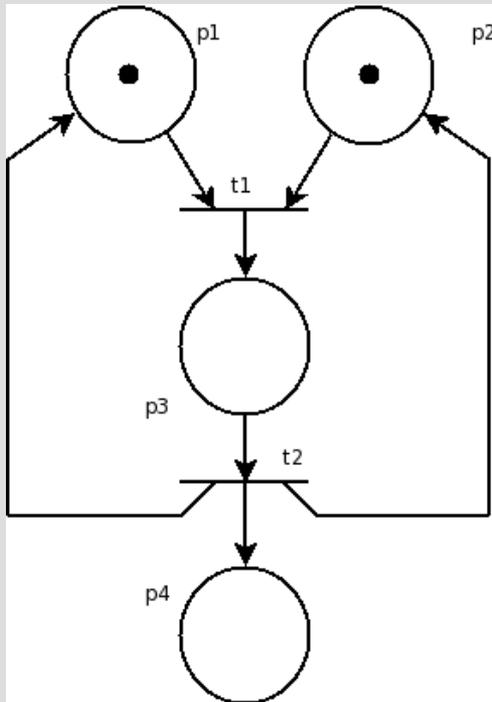
A raiz da árvore é o estado correspondente à marcação inicial da árvore.

Espaço de estados finito;
A árvore contém todos os estados
e respectivas transições



A árvore de cobertura (*coverability tree*)

- **Nem sempre** as redes de Petri apresentam espaço de estados **finito**.
 - Árvore de cobertura de PN com espaço de estados **infinito**
 - O símbolo ω representa um **número ilimitado de *tokens***



A árvore de cobertura: Definição Formal

(*coverability tree*)

Def. Seja a rede de Petri $N=(P,T,A,w,\underline{x}_0)$,

- A **árvore de cobertura** é uma árvore em que os **ramos** denotam **transições** $t \in T$ e os **nós** representam **estados ω -estendidos** da rede de Petri.
- Um **estado ω -estendido** \underline{x} consiste num vector de marcação $\underline{x} \in (\mathbb{N}^{|\mathcal{P}|} \times \omega)^{|\mathcal{P}|}$ o qual representa **todos** os estados da rede que podem ser **derivados** substituindo ω por um número natural $n \in \mathbb{N}$.

A árvore de cobertura: Definição Formal

(*coverability tree*)

Def. (cont.)

- Diz-se que um certo estado **x cobre o estado y** se:
 $\forall p \in P : x(p) = \omega \text{ ou } \{x(p) \neq \omega \text{ e } x(p) \geq y(p)\}$
- A **nó terminal** é um estado ω -estendido em que no qual **não há transições permitidas**
- A **nó duplicado** é um estado ω -estendido que **já existe** algures na árvore de cobertura

Árvore de cobertura: algoritmo de construção

Algoritmo Karp-Miller

1. Label initial marking M_0 as the root of the tree and tag it as new
2. While new markings exist do:
 - select a new marking M
 - if M is identical to a marking on the path from the root to M , then tag M as old and go to another new marking
 - if no transitions are enabled at M , tag M dead-end
 - while there exist enabled transitions at M do:
 - obtain the marking M' that results from firing t at M
 - on the path from the root to M if there exists a marking M'' such that $M'(p) \geq M''(p)$ for each place p and M' is different from M'' , then replace $M'(p)$ by ω for each p such that $M'(p) > M''(p)$
 - introduce M' as a node, draw an arc with label t from M to M' and tag M' as new.

Análise de redes de Petri

A árvore de cobertura permite responder a diversas das questões formuladas inicialmente!

Limitação e segurança (*boundeness, safeness*)

- Os lugares representam vulgarmente recursos físicos como *buffers*, filas, etc. Se na árvore de cobertura existe pelo menos **um estado** em que o símbolo ω aparece então a rede é **não limitada**.
- Se o lugar corresponder fisicamente a um certo recurso físico limitado o sistema **não é seguro** pois o recurso em causa pode ser **esgotado**!

Análise de redes de Petri

Conservação (*conservation*)

Em muitos sistemas é necessário verificar a propriedade da **conservação**, i.e., se o número de *tokens* se mantém constante. E.g. se os tokens representarem impressoras, é importante verificar que, após a submissão de trabalhos as impressoras são libertadas.

- Uma rede de Petri é **conservativa em sentido estrito** se para todos os estados alcançáveis o número de tokens é constante

$$\sum_{p \in P} y(p) = \sum_{p \in P} x_0(p)$$

- Esta propriedade pode ser tomada também em **sentido lato**, quando apenas um conjunto de lugares contribuem para a propriedade da conservação. Neste caso a equação acima aplica-se apenas a esse conjunto.

Análise de redes de Petri

Cobertura (*coverability*)

A simples inspecção visual da árvore de cobertura permite determinar a cobertura. Há algoritmos eficientes para cálculo de sequência de transições mais curta que leva a um estado que cobre outro

- Nota: diz-se que um estado \underline{x} cobre um estado \underline{y} se pelo menos todas as transições permitidas em \underline{y} o são também em \underline{x} , ou seja:
 - $\forall p \in P \ x(p) \geq y(p)$

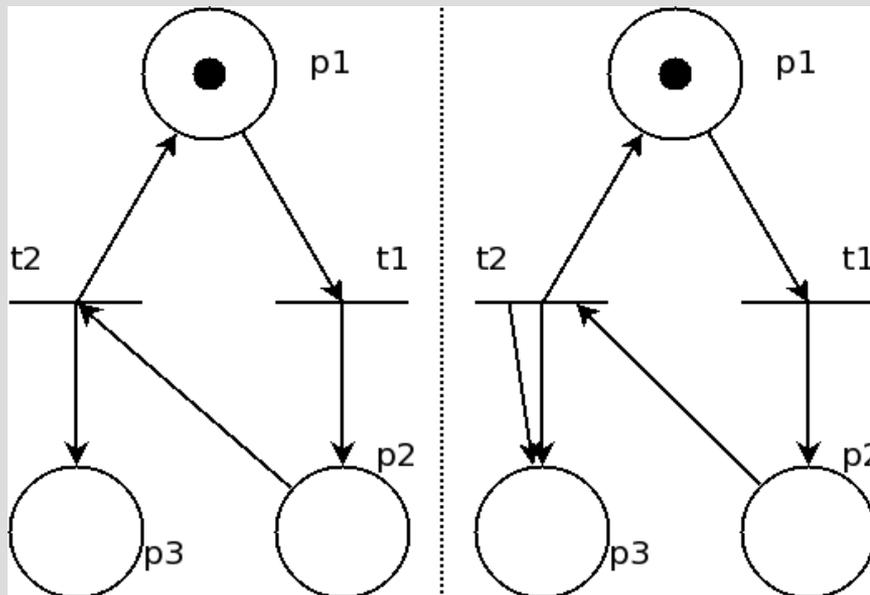
Limitações

No caso geral a árvore de cobertura **não permite determinar o conjunto de estados alcançáveis**, o que acaba por limitar a efectividade desta ferramenta na detecção de *deadlocks*, *liveness*, etc.

Análise de redes de Petri

Limitações (cont)

Considere-se as seguintes redes de Petri:



$[1, 0, 0]$

$t_1 \downarrow$

$[0, 1, 0]$

$t_2 \downarrow$

$[1, 0, \omega]$

$t_1 \downarrow$

$[0, 1, \omega]$

$t_2 \downarrow$

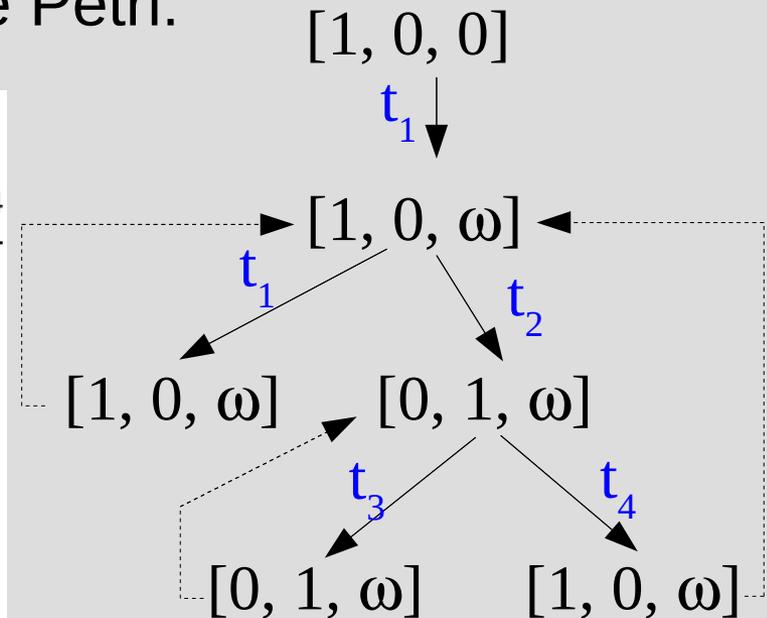
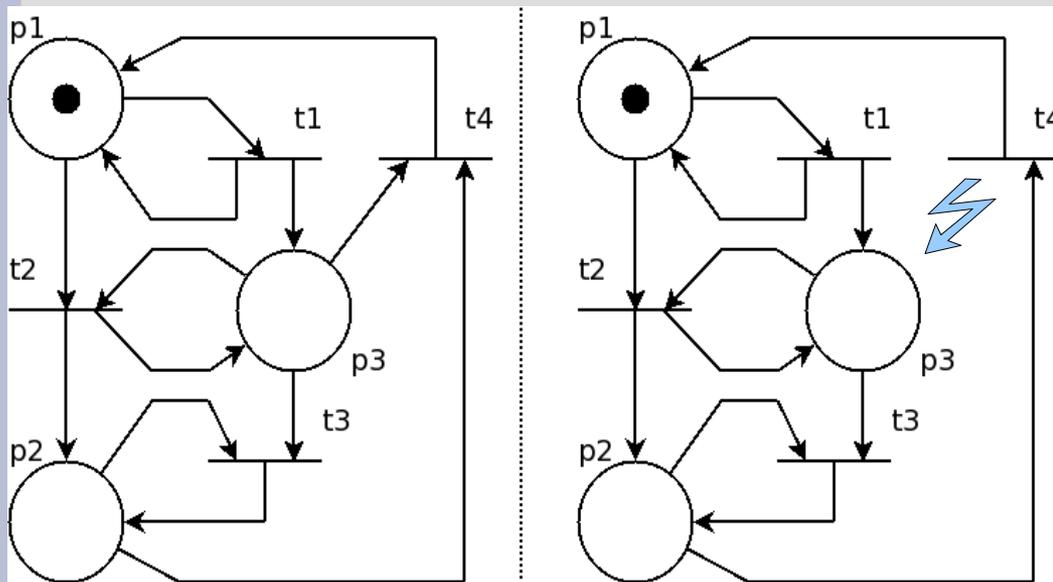
$[1, 0, \omega]$

- Duas redes diferentes que têm **a mesma árvore de cobertura**
- Na árvore à **esquerda** p3 pode ter qualquer número **inteiro** de tokens
- Na árvore à **direita** p3 tem sempre um **número par** de tokens

Análise de redes de Petri

Limitações (cont)

Considere-se as seguintes redes de Petri:



Duas redes diferentes que têm **a mesma árvore de cobertura**

- A árvore à **esquerda** pode entrar em deadlock (e.g. t_1, t_2, t_3)
- Na árvore à **direita** não há deadlocks

Análise de redes de Petri

Limitações (cont)

Apesar destas limitações, a árvore de cobertura permite **resolver** muitas situações práticas, e.g.:

- Se um estado **sem ω** aparece na árvore de cobertura então é **alcançável**
- Se um estado **não aparece** na árvore de cobertura então **não é alcançável**
- Assim, as redes de Petri permitem, em **muitas situações práticas** responder a problemas de alcançabilidade, *liveness* e *deadlock*!

Subclasses de redes de Petri

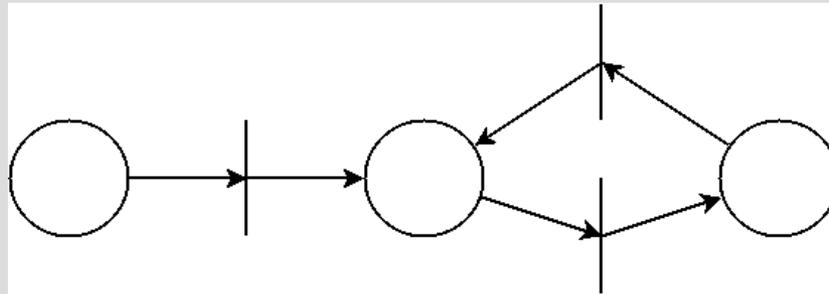
Introduzindo algumas restrições pode **aumentar-se a capacidade de análise** efectiva das redes de Petri à custa de uma **redução** da sua **expressividade**

- Subclasses de rede de Petri:
 - **Máquina de Estados**
 - **Grafos Marcados** (*Marked graphs*)

Subclasses de redes de Petri

Máquina de Estados

- Cada **transição** tem no máximo **um antecessor** e **um sucessor**
 - Apenas modela **causalidade** e **conflito**
 - **Não** modela **concorrência** nem **sincronização** de actividades paralelas

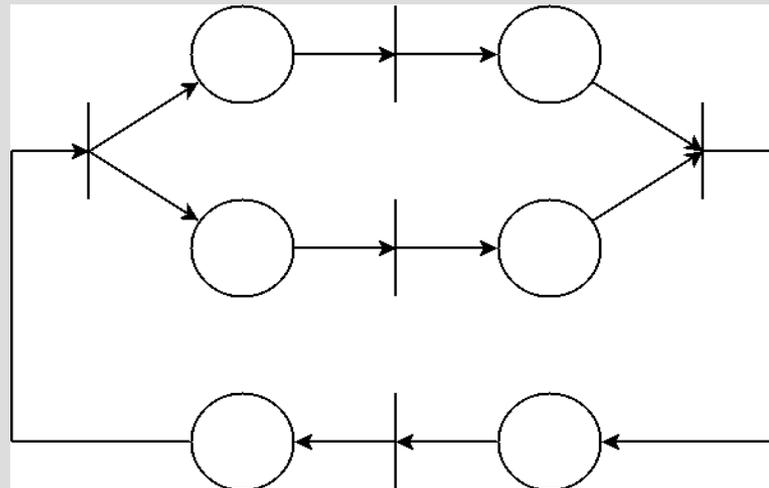


- O espaço de estado é finito
- Conservativas em sentido estrito

Subclasses de redes de Petri

Grafos marcados

- Cada **lugar** tem no no máximo **um antecessor** e **um sucessor**
 - Modela **concorrência**
 - **Não** modela **causalidade** nem **conflito**



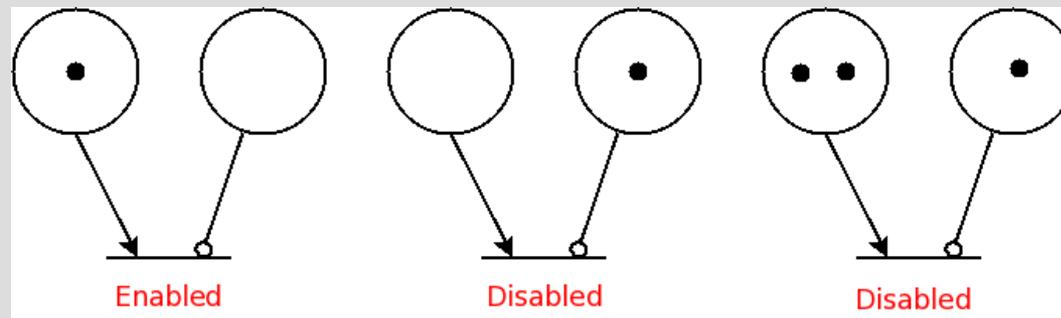
- Modelo equivalente ao SDF

Extensões às redes de Petri

Existem também **extensões** às redes de Petri que permitem **aumentar** a sua **expressividade**

▪ Arcos inibidores

- Um arco inibidor **modifica** a regra de **disparo**, por forma a que uma transição é permitida apenas quando **existem** *tokens* em todos os seus **lugares** de entrada **normais** e **não há** *tokens* nos **lugares** ligados por **ramos inibidores**.



Extensões às redes de Petri

▪ Timed Petri Nets

Noção de **tempo** pode ser associada a **lugares**, **transições** ou mesmo **arcos**

▪ Tempo associado a lugares

- Um *token* **chega** a um lugar num instante t e apenas fica **disponível** para transições de saída **após** um tempo $t+d$. (tempo d específico a cada lugar)

▪ Tempo associado a transições

- Cada **transição** tem associados **dois valores** τ_1 e τ_2 .
Uma transição apenas pode disparar se estiver **enabled durante** τ_1 e tem de disparar antes do tempo τ_2 ter **decorrido** (define **janela** de disparo)

Extensões às redes de Petri

▪ Timed Petri Nets (cont.)

▪ Tempo associado a arcos

- Um **arco** temporal que liga um lugar “p” a uma transição “t” tem **associado** um **intervalo** $[d_1, d_2]$.
- Quando um *token* **chega** a “p” no instante τ o arco **contribui** para o **disparo** da transição “t” apenas durante o **intervalo** $[\tau+d_1, \tau+d_2]$.
- Se o token **não** for **consumido** nesse intervalo **não** poderá mais ser consumido por essa transição.

Nota: há muitas outras variantes de Timed Petri Nets.

Extensões às redes de Petri

▪ Redes de Petri Coloridas

- ♦ Cada **token** tem associada um **tipo** ou **cor**
- ♦ Os **lugares** são associados a **tipos**, os quais **restringem** o **tipo** de *tokens* que podem **receber**
- ♦ Quando uma transição **dispara** os *tokens* consumidos são **transformados** em *tokens* eventualmente de **outros tipos/cores**
- ♦ **Tipo de rede de Petri muito usado na actualidade!!!**

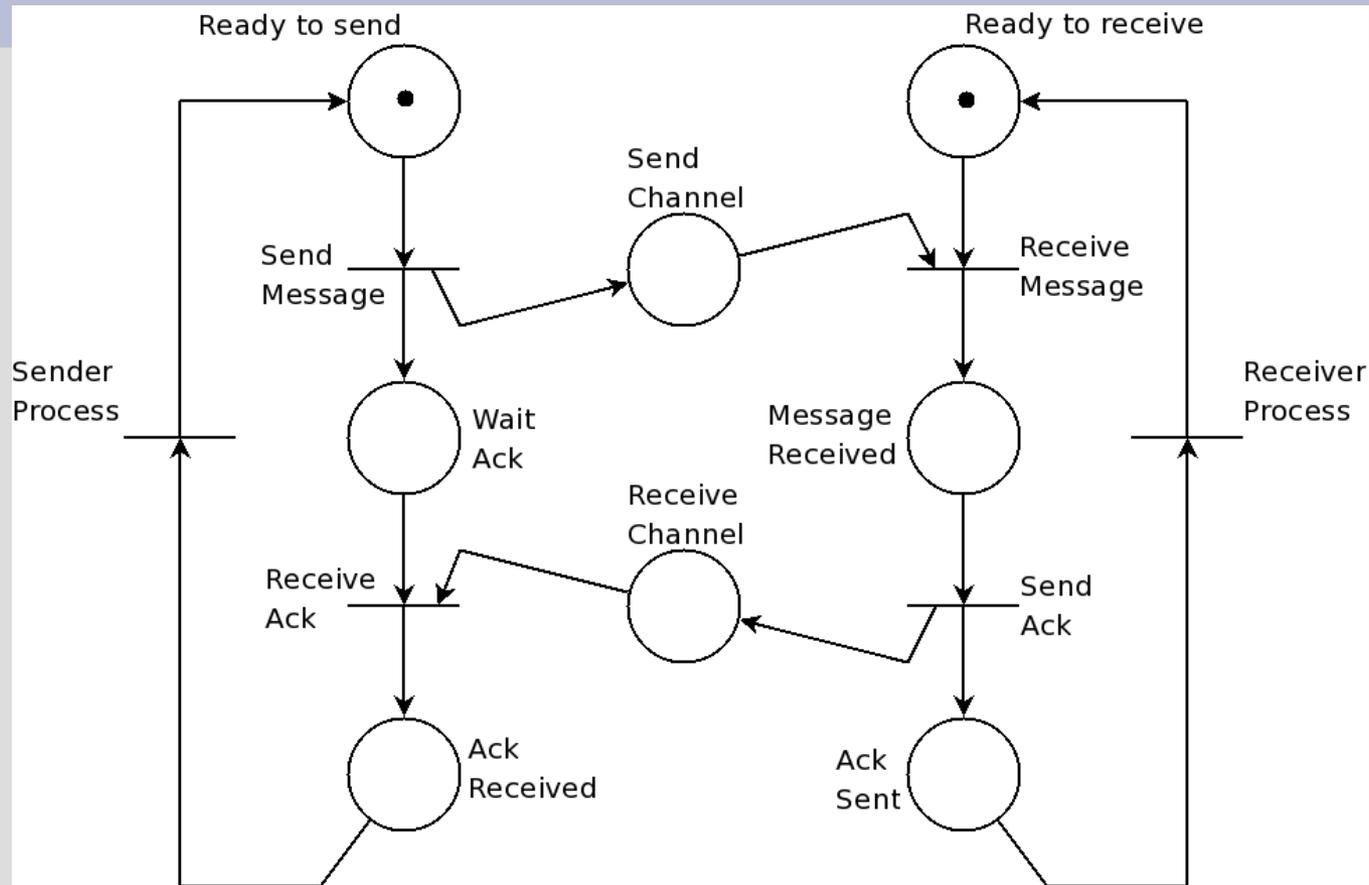
Muitas outras extensões foram desenvolvidas

Exercício

Exercício: modelar, por meio de uma rede de Petri, a operação de um protocolo de comunicação do tipo “Send/Wait”

- Dois processos, um **produtor** e outro **consumidor**
- Processo **produtor envia** uma mensagem e tem de **aguardar** por um *acknowledge*. Após receber o *acknowledge* pode efectuar processamento e posteriormente enviará uma nova mensagem
- Processo **consumidor aguarda** que uma mensagem chegue e, quando tal acontece, **envia** um *acknowledge*, processa a mensagem e fica pronto a receber uma nova mensagem

Exercício



- Calcular a árvore de cobertura
- Modificar para o caso de perda de mensagens
 - Usar *Timed Petri Nets* para especificar *timeouts*

Alguns links ...

- Existem imensos recursos na Internet sobre Petri Nets, versando conceitos básicos, extensões, subclasses, ferramentas de análise e simulação, etc.
- Um excelente repositório de informação encontra-se em:
<http://www.informatik.uni-hamburg.de/TGI/PetriNets/>
- Este site inclui também um excelente levantamento de ferramentas de simulação:
<http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html>

Sumário

▪ Conceitos básicos

- ♦ Definição
- ♦ Função de transição
- ♦ Conjunto de alcançabilidade
- ♦ Modelação de I/O
- ♦ Templates

▪ Análise

- ♦ Matriz de incidência
- ♦ Árvore de cobertura

▪ Subclasses

- ♦ Máquinas de Estados
- ♦ Grafos Marcados

▪ Extensões

- ♦ Arcos inibidores
- ♦ Timed Petri Nets
- ♦ PN coloridas