

Sistemas Operacionais

Comunicação entre processos

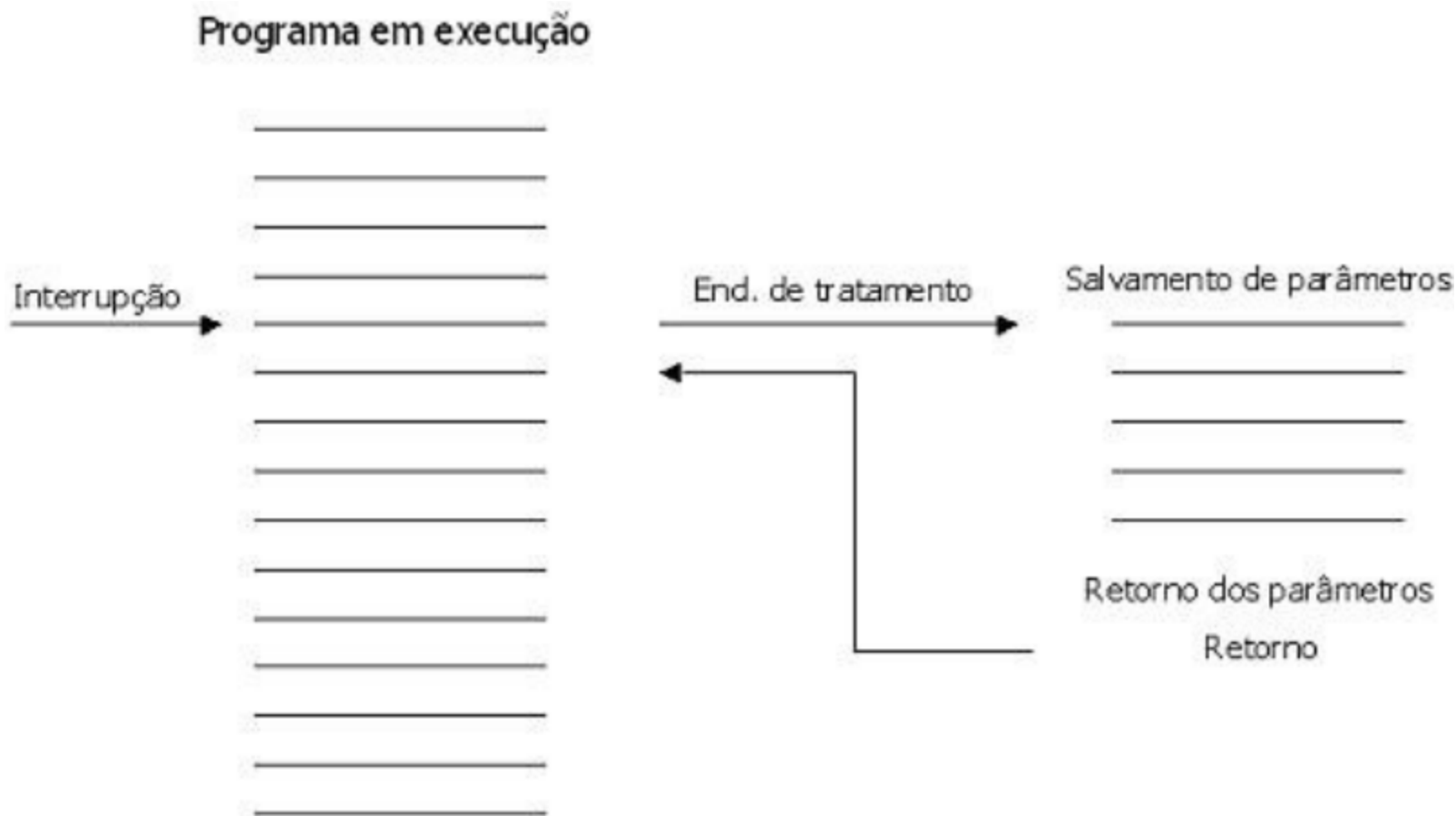
Comunicação entre Processos

- Num sistema de multiprocessamento ou multiprogramação, os processos geralmente precisam se comunicar com outros processos.
- A comunicação entre processos é mais eficiente se for estruturada e não utilizar interrupções.

Comunicação entre Processos

- O que são interrupções?
- Uma interrupção é um evento externo que faz com que o processador pare a execução do programa corrente e desvie a execução para um bloco de código chamado rotina de interrupção (normalmente são decorrentes de operações de E/S).
- Ao terminar o tratamento de interrupção o controle retorna ao programa interrompido exatamente no mesmo estado em que estava quando ocorreu a interrupção.

Comunicação entre Processos



Comunicação entre Processos

- Condição de Disputa ;
- Região Crítica;
- Formas de Exclusão Mútua ;
- Problemas Clássicos.

Comunicação entre Processos

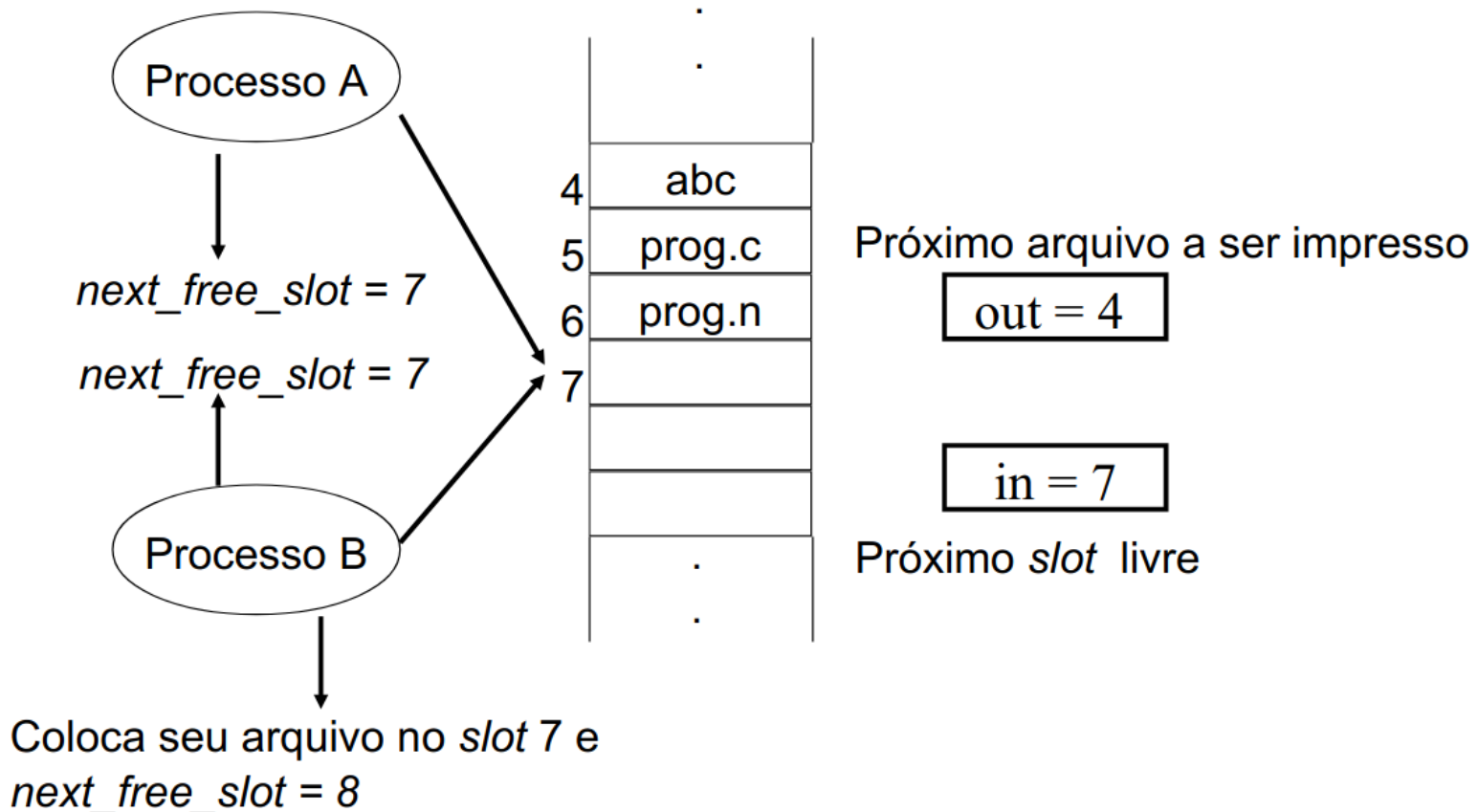
- Processos competem por recursos.
- Três aspectos importantes:
 - Como um processo passa informação para outro processo;
 - Como garantir que processos não invadam espaços uns dos outros;
 - Dependência entre processos: sequência adequada.

Condições de Disputa (*Race Conditions*):

- Processos acessam recursos compartilhados concorrentemente.
- Recursos: memória, arquivos, impressoras, discos, variáveis.
 - Impressão: quando um processo deseja imprimir um arquivo, ele coloca o arquivo em um local especial chamado diretório de *spool* (*spooler*);
 - Um outro processo, chamado *daemon* de impressão, checa se existe algum arquivo a ser impresso. Se existe, esse arquivo é impresso e retirado do *spooler*. Imagine dois processos que desejam imprimir um arquivo ao mesmo tempo.

Condições de Disputa (*Race Conditions*):

Spooler – fila de impressão (*slots*)



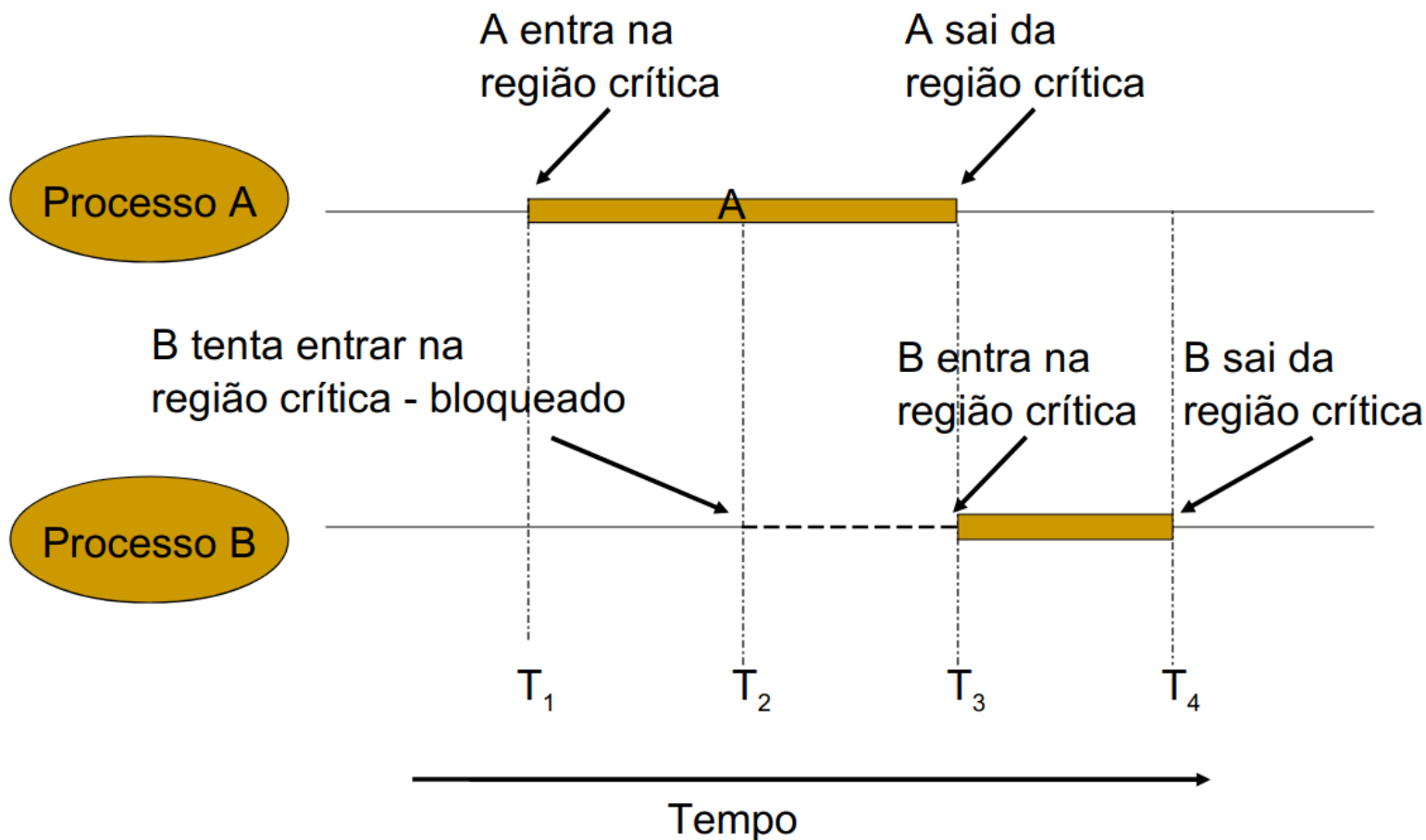
Condições de Disputa (*Race Conditions*):

- Como solucionar problemas de *Condições de Disputa*???
- Proibir que mais de um processo leia ou escreva em recursos compartilhados concorrentemente (ao “mesmo tempo”)
- **Recursos compartilhados** = Regiões Críticas.
- **Exclusão mútua**: modo de garantir que outros processos sejam impedidos de usar uma variável ou um arquivo compartilhado que já esteja em uso por um processo.


Condições de Disputa (*Race Conditions*):

- Quatro condições para uma boa solução:
 1. Dois processos não podem estar simultaneamente em suas regiões críticas;
 2. Nenhuma restrição deve ser feita com relação à CPU;
 3. Processos que não estão em regiões críticas não podem bloquear outros processos que desejam utilizar regiões críticas;
 4. Processos não podem esperar para sempre para acessarem suas regiões críticas.

Condições de Disputa (*Race Conditions*):




Condições de Disputa (*Race Conditions*):



Thread 1	Thread 2		Integer value
			0
read value		←	0
	read value	←	0
increase value			0
	increase value		0
write back		→	1
	write back	→	1



Condições de Disputa (*Race Conditions*):

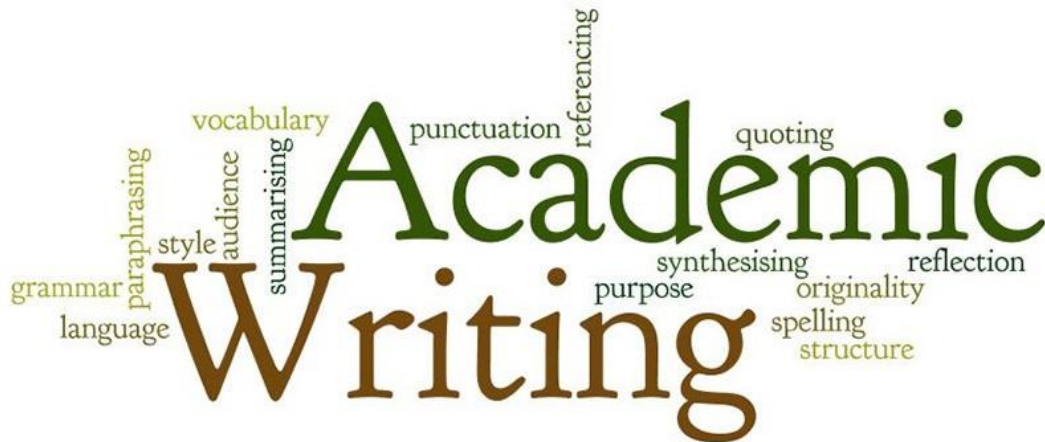


Thread 1	Thread 2		Integer value
			0
read value		←	0
	read value	←	0
increase value			0
	increase value		0
write back		→	1
	write back	→	1

Thread 1	Thread 2		Integer value
			0
read value		←	0
increase value			0
write back		→	1
	read value	←	1
	increase value		1
	write back	→	2



- Data Races vs. Data Race Bugs: Telling the Difference with Portend
- Hybrid Dynamic Data Race Detection



Modelagem de concorrência: Petri Net

Classificação dos Modelos

- Modelos Baseados em Estado
 - Consideram apenas o conjunto **S** para modelar e se referir a propriedades do sistema.
 - Maioria das lógicas temporais: CTL (**Computation Tree Logic**)
- Modelos Baseados em Ações
 - Consideram apenas o conjunto **T** para modelar e se referir a propriedades dos sistemas.
 - As álgebras de processos: CCS, CSP, FSP
- Modelos Mistos
 - Consideram ambos os conjuntos **S** e **T**.
 - Redes de Petri

Modelagem de concorrência: Petri Net

Redes de Petri

- Áreas de Aplicação:
 - Concorrência
 - Arquitetura de Computadores
 - Protocolo de Redes
 - Sistemas Operacionais
 - Sistemas de Produção
 - Sistemas Digitais
 - *Hardware/Software Co-design*
 - Engenharia de Software
 - Sistemas de Tempo Real
 - Modelagem e Avaliação de Desempenho
 - Diagnóstico de Falhas
 - Controle de Tráfego
 - *Workflow*
 - Administração
 - Química
 - etc

Redes de Petri

Componentes

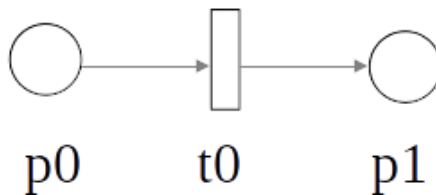


Lugar



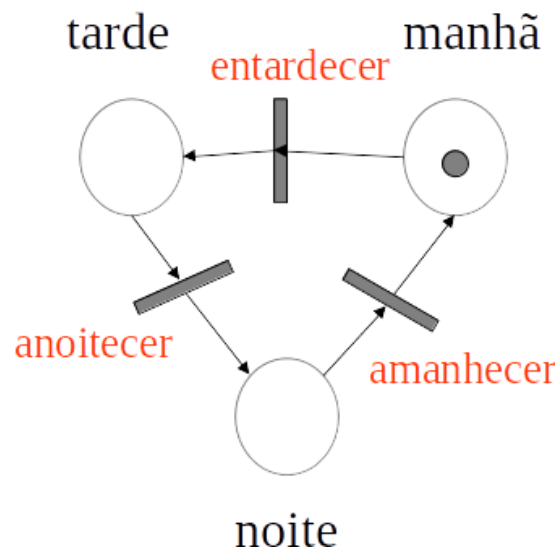
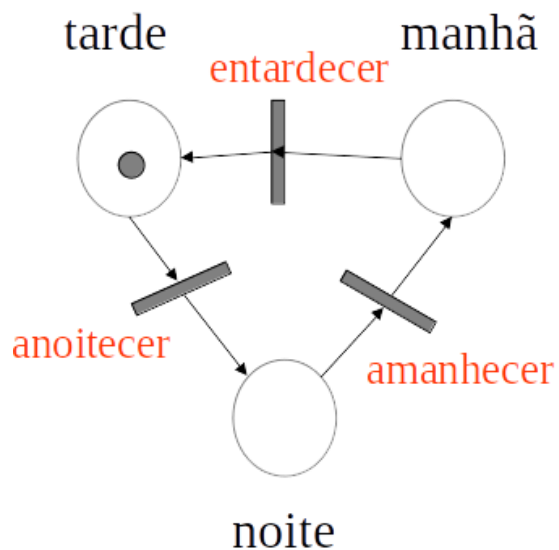
Transição

Rede



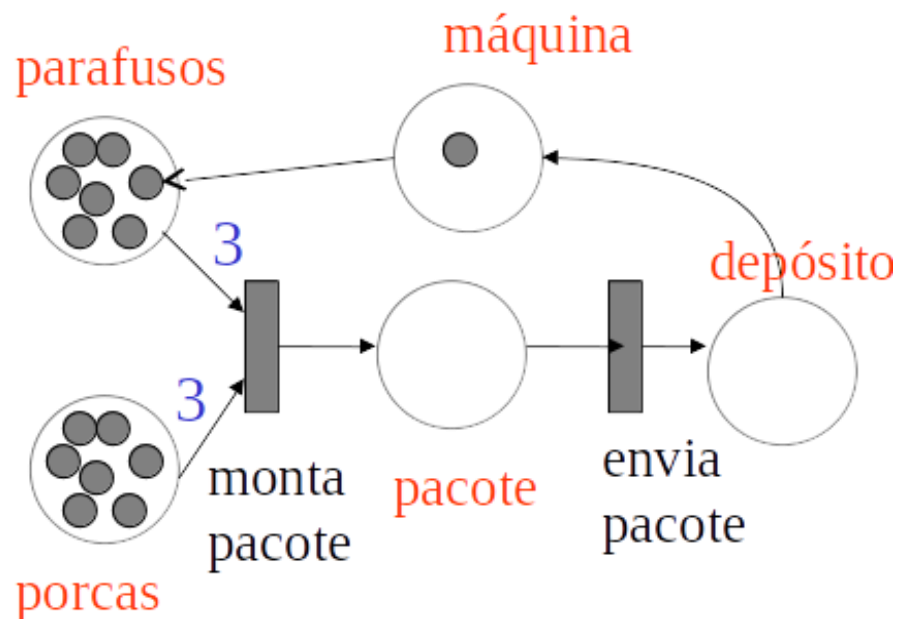
Redes de Petri

- Períodos do Dia



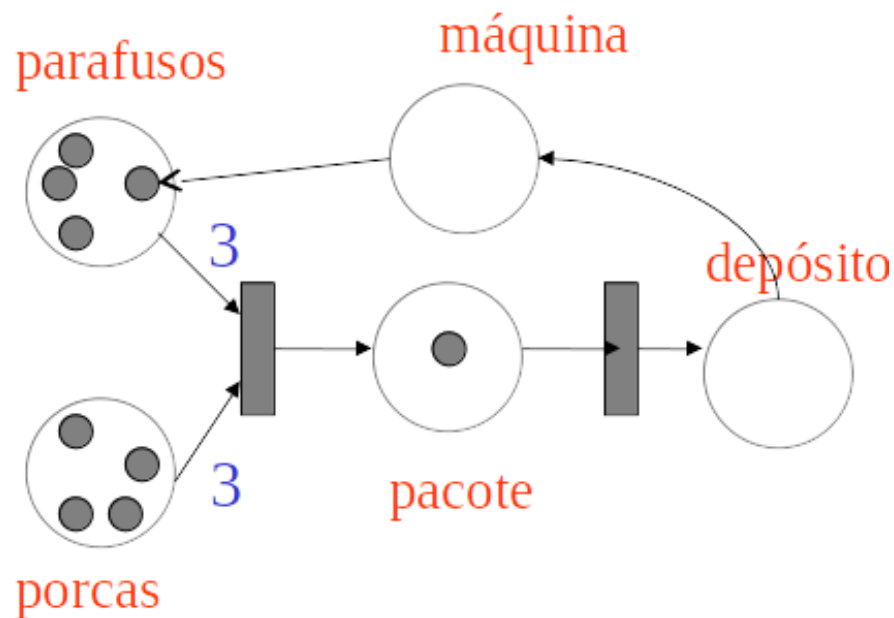
Redes de Petri

- Linha de Produção



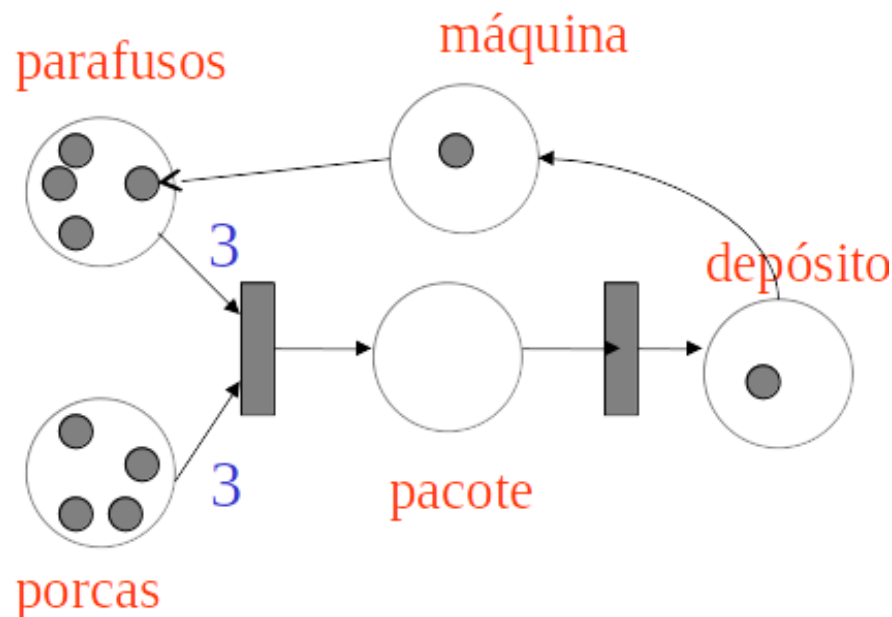
Redes de Petri

- Linha de Produção



Redes de Petri

- Linha de Produção



Redes de Petri

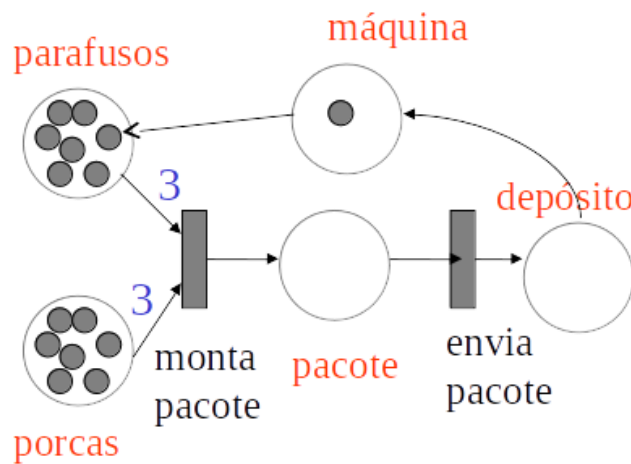
- Definição: *Place/Transition Nets* - Teoria *Bag* (multiconjuntos)

$$R = (P, T, I, O, M_0)$$

- P - Conjunto de lugares - $P = \{p_0, \dots, p_n\}$
- T - Conjunto de transições - $T = \{t_0, \dots, t_m\}$
- I - Conjunto de *bags* de entrada - $I: T \rightarrow P^\infty$, é um conjunto de ***bags*** que representa o mapeamento de transições para lugares de entrada.
- O - Conjunto de *bags* de saída - $O: T \rightarrow P^\infty$
- M_0 - Vetor marcação inicial - $M_0: P \rightarrow \mathbb{N}$

Redes de Petri

- Linha de Produção



$$M_0 = |7, 7, 0, 1, 0|$$

- $R_{LP} = (P, T, I, O, M_0)$

$P = \{ \text{parafusos, porcas, pacote, máquina depósito} \}$

$T = \{ \text{monta_pacote, envia_pacote} \}$

$I = \{ I(\text{monta_pacote}), I(\text{envia_pacote}) \}$

$O = \{ O(\text{monta_pacote}), O(\text{envia_pacote}) \}$

$I(\text{monta_pacote}) = [\text{parafusos, parafusos, parafusos, porcas, porcas, porcas, máquina}]$

$I(\text{envia_pacote}) = [\text{pacote}]$

$O(\text{monta_pacote}) = [\text{pacote}]$

$O(\text{envia_pacote}) = [\text{máquina, depósito}]$

Tools



Visual C++ Redistributable for Visual Studio 2015

-Para instalar o Snoopy

<https://www.microsoft.com/en-us/download/details.aspx?id=48145>

Snoopy

<http://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Snoopy>

INA

<http://www2.informatik.hu-berlin.de/lehrstuehle/automaten/ina/>

Mais sobre Redes de Petri

<http://disciplinas.stoa.usp.br/course/view.php?id=3078>

Exemplo ATM

- Time = t_0 Authorisation OK
- Time = $t_0 + 2s$ Balance is 1000
- Time = $t_0 + 10s$ Client requested 800, $800 < 1000$
- Time = $t_0 + 12s$ Account updated by -800, 200 left
- Time = $t_0 + 14s$ Cash dispensed



Exemplo ATM



Saque da mesma conta ao mesmo tempo?

Exemplo ATM

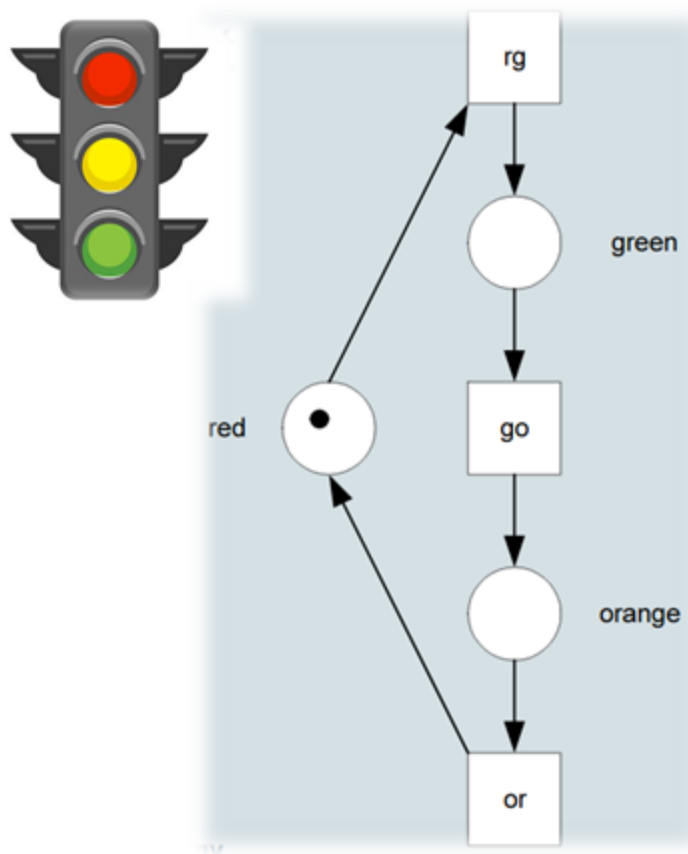
	Client 1	Client 2
Time = t_0	Authorisation OK	
Time = t_0+1s		Authorisation OK
Time = t_0+2s	Balance is 1000	
Time = t_0+3s		Balance is 1000
Time = t_0+5s		Client requested 800, $800 < 1000$
Time = t_0+7s		Account updated by -800, 200 left
Time = t_0+9s		Cash dispensed
Time = t_0+10s	Client requested 800, $800 < 1000$	
Time = t_0+12s	Account updated by -800, -600 left	
Time = t_0+14s	Cash dispensed	

Exemplo ATM

	Client 1	Client 2
Time = $t_0 + 2s$	Balance is 1000, lock account	
Time = $t_0 + 3s$		account locked - cannot proceed
Time = $t_0 + 10s$	Client requested 800, $800 < 1000$	
Time = $t_0 + 12s$	Account updated by -800, 200 left, unlock account	
Time = $t_0 + 14s$	Cash dispensed	



Exemplo Sinalização



Exemplo Sinalização

