PMR 5237

Modelagem e Design de Sistemas Discretos em Redes de Petri

Aula 10 : Modelagem de sistemas com RdP

Prof. José Reinaldo Silva
reinaldo@usp.br

1) *boundedness*, characterising finiteness of the state space.
2) *liveness*, related to potential fireability in all reachable markings. *Deadlock-freeness* is a weaker condition in which global infinite activity (i.e. fireability) of the net system model is guaranteed, but some parts of it may not work at all.
3) *reversibility*, characterising recoverability of the initial marking from any reachable marking.
4) *mutual exclusion*, dealing with the impossibility of simultaneous *submarkings* (p-mutex) or *firing concurrency* (t-mutex).

0) *dead (L0-live)* if t can never be fired in any firing sequence in $L(M_0)$.
1) *L1-live (potentially firable)* if t can be fired at least once in some firing sequence in $L(M_0)$.
2) *L2-live* if, given any positive integer k, t can be fired at least k times in some firing sequence in $L(M_0)$.
3) *L3-live* if t appears infinitely, often in some firing sequence in $L(M_0)$.
4) *L4-live* or *live* if t is *L1*-live for every marking M in $R(M_0)$.

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.

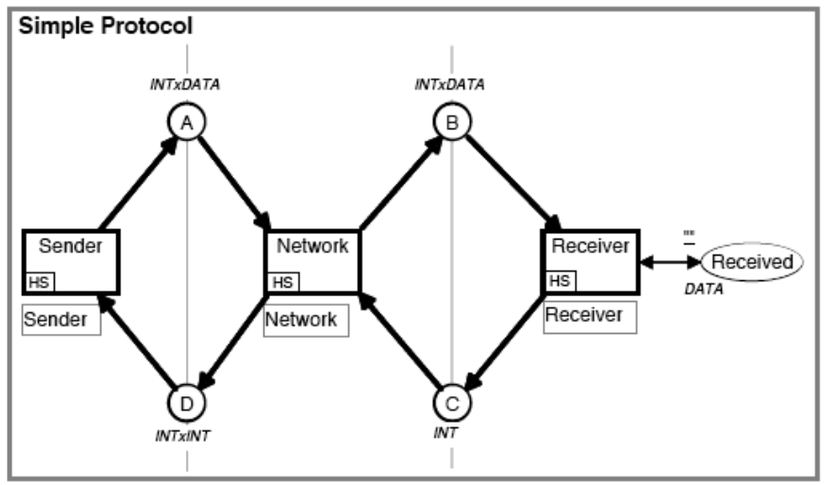
Most of the industrial applications would be *totally impossible* without:

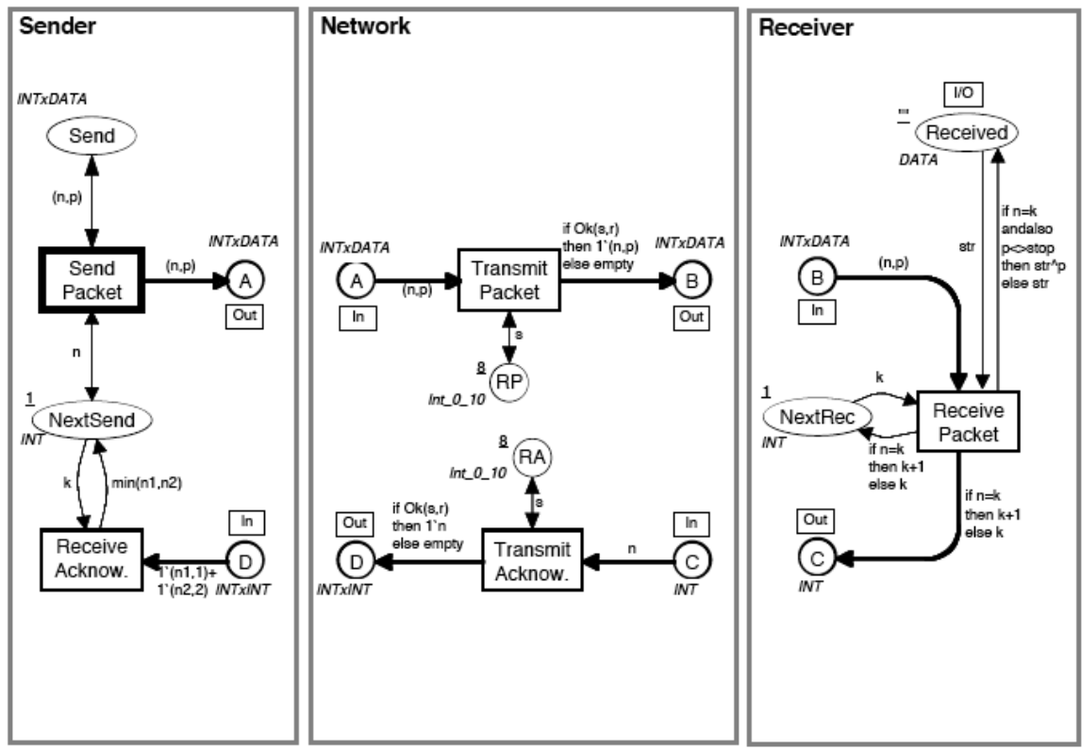
- Colours.
- Hierarchies.
- Computer tools.

A 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.

A hierarchical CP-net contains a number of *interrelated subnets*– called *pages*.

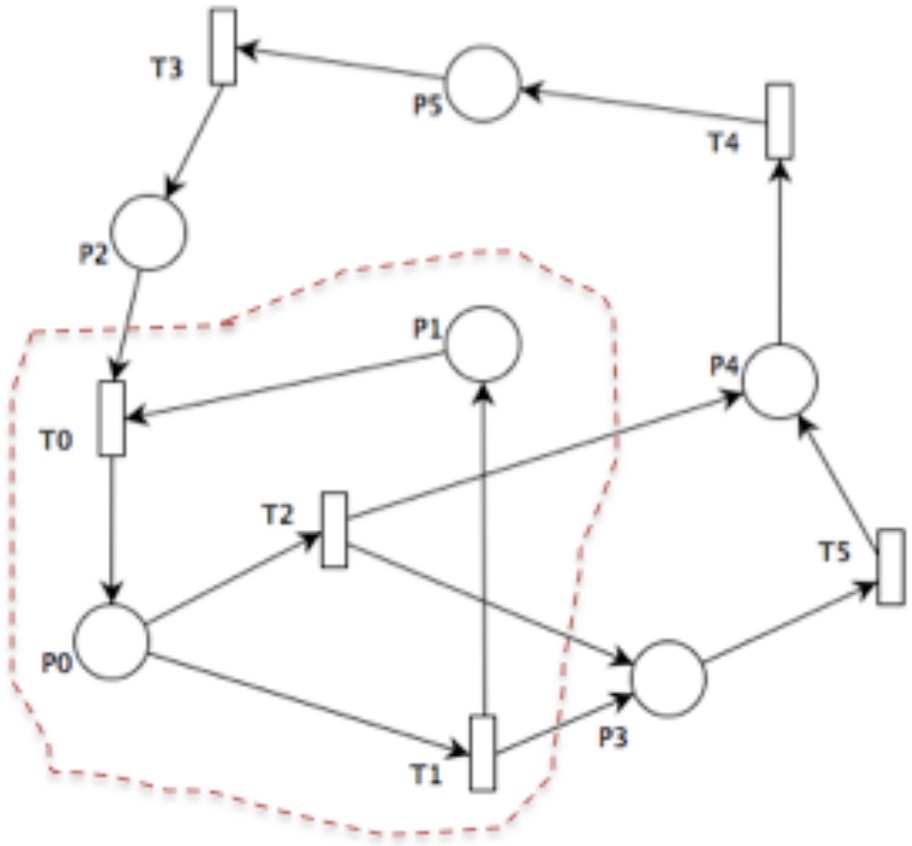




Hierarchy is not anything new and is actually connected with any kind of net, including the classical ones.

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

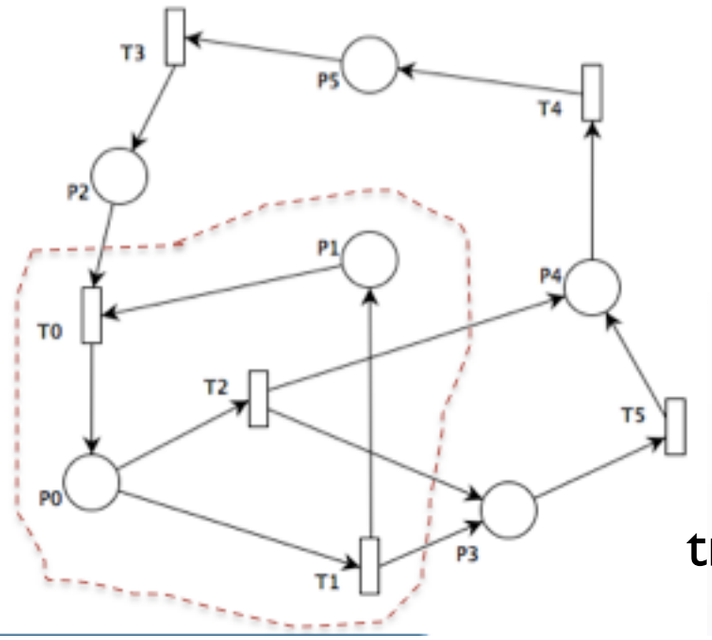
Hierarquia em redes clássicas

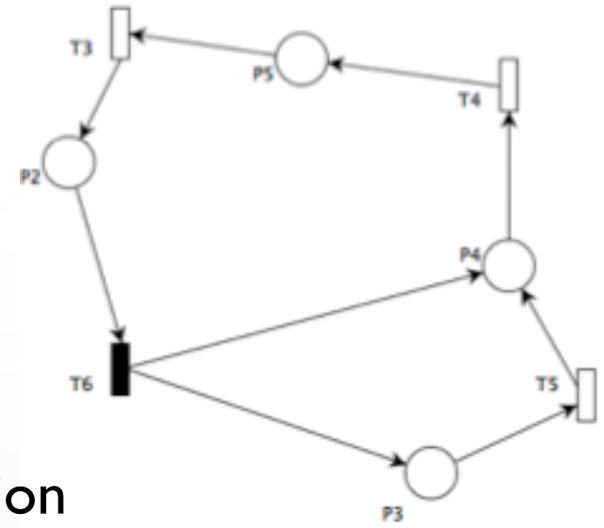


Definition 39

Seja uma estrutura de rede $N = (S, T; F)$. Seja $X = S \cup T$ e um sub-cojunto $Y \subseteq X$. Definimos uma borda de N como o conjunto $\partial(N) = \{y \in Y \,|\, \exists x \notin Y . x \in loc(y)\}$.

Substituição de uma sub-rede

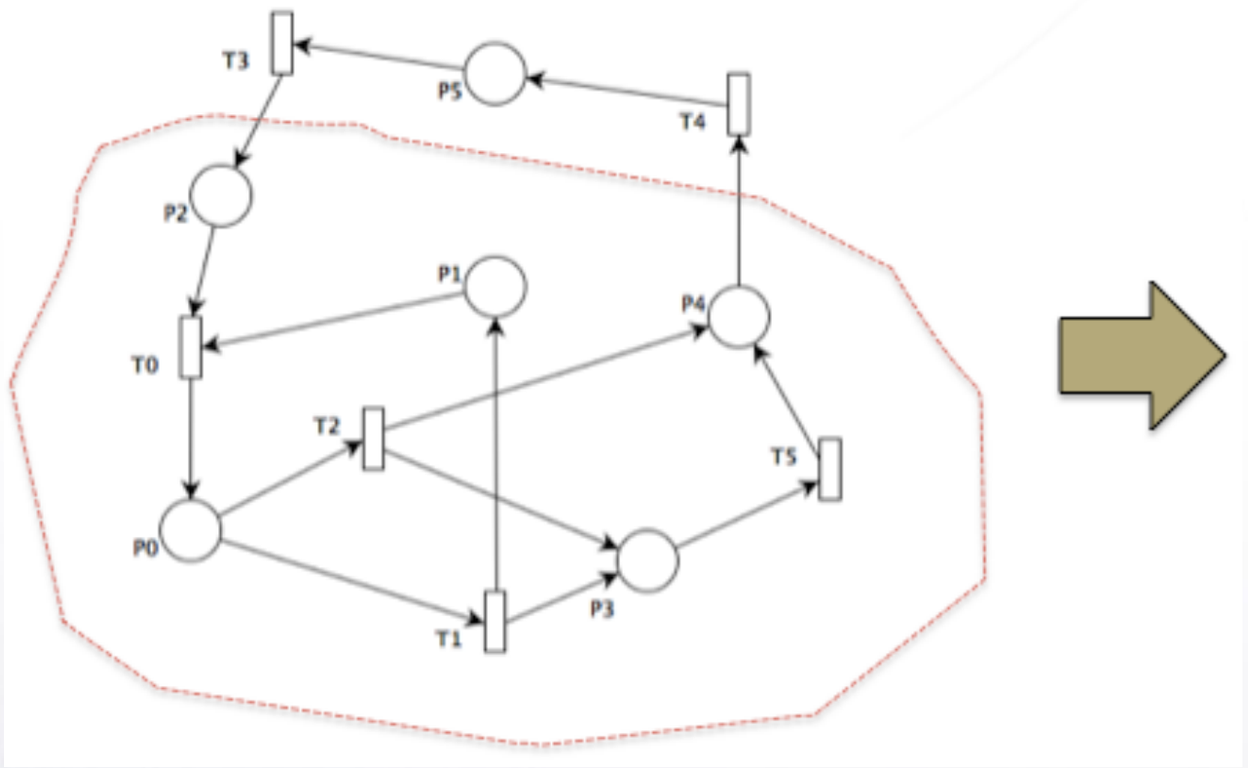


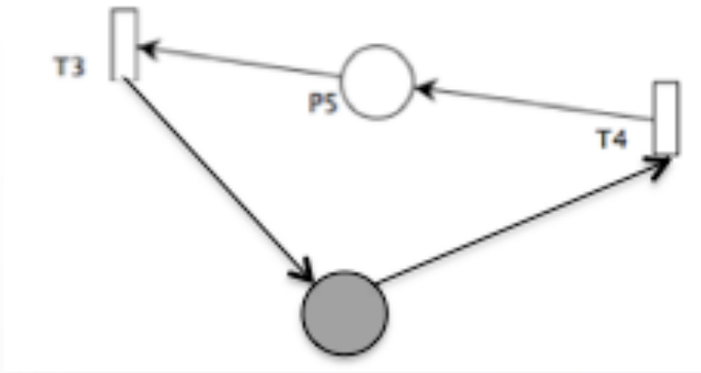


transition bounded substitution

Definition 40

Um sub-conjunto de elementos Y da rede $N = (S, T; F)$ é dito limitado por lugar (place bounded) ou aberto, se e somente se $\partial(Y) \subseteq S$. Similarmente, um sub-conjunto Y desta rede é dito limitado por transição (transition bounded), se e somente se $\partial(Y) \subseteq T$.





place bounded substitution

Se em uma rede com estrutura $N = (S, T; F)$ existe uma sub-rede Y limitada por transição, a substituição desta sub-rede Y gera uma rede $N' = (S', T'; F')$ onde:

(i) $S' = S \setminus Y$;

(ii) $T' = T \setminus Y \cup \{t_y\}$, onde t_y é o novo elemento que substitui Y;

(iii) $F' = F \setminus Int(Y)$ onde $Int(Y)$ é o conjunto dos arcos internos de Y.

.

Similarmente, se a sub-rede Y é limitada por lugar,

(i) $S' = S \setminus Y \cup \{s_y\}$, onde s_y é o novo elemento que substitui Y;

(ii) $T' = T \setminus Y$;

(iii) $F' = F \setminus Int(Y)$ onde $Int(Y)$ é o conjunto dos arcos internos de Y.

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.

Elementos próprios

Seja x_y um elemento genérico (instanciável por t_y ou por p_y). Este elemento é dito *próprio* se e somente se é limitado por transição (lugar), tem somente dois elementos de borda, com pelo menos um processo vivo entre eles.

Se os elementos abstratos são próprios as propriedades da rede subjacente se conservam a menos de um termo aditivo. (J. R. Silva, On The Property Analysis of Abstract and Hierarchical Nets, to appear).

Fundamentos do método estruturado

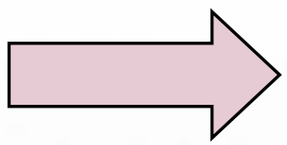
Um ***bloco*** é um conjunto genérico de instruções de programa, onde uma dada instrução é identificada como a <u>entrada</u> do bloco e outra (diferente da primeira) é identificada como a <u>saída</u>.

Se A e B são blocos de um mesmo programa, então A e B são ditos independentes se e somente se $A \cap B = \varnothing$.

Se A e B são tais que $A \cap B \neq \varnothing$ então $(A \subseteq B)$ OU $(A \supseteq B)$

Constituintes próprios e primos

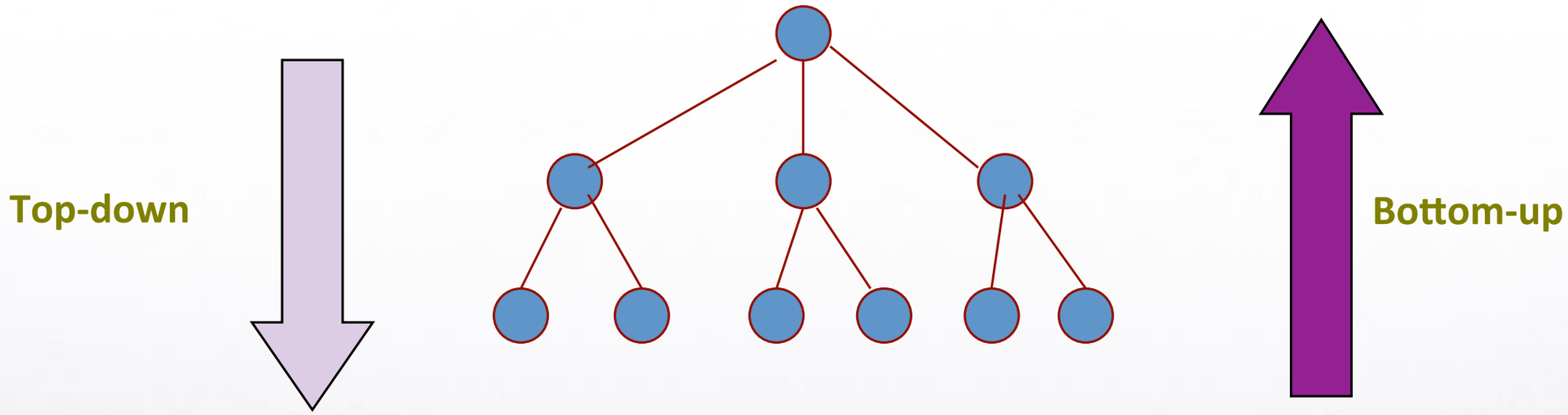
Elementos próprios



Base de elementos constituintes (componentes)

Elementos próprios indivisíveis são chamados primos. Um conjunto LI de elementos primos pode constituir uma base e portanto pode descrever qualquer programa.

Decomposição por refinamentos: o método estruturado



Overview

Apresentamos as redes de Petri como um esquema e uma representação formal para a modelagem e análise de sistemas e processos discretos, pertinentes a uma larga gama de domínios. Em particular, quase 70% dos sistemas automatizados acabam caindo nesta categoria, e o futuro nos reserva ainda possibilidade de ampliação deste escopo com a difusão dos "sistemas de serviço".

O importante é no entanto a introdução do formalismo de redes de Petri, que como vimos pode ser dividido em dois grandes grupos: o das redes ditas clássicas; o das redes de alto nível ou redes estendidas temporizadas ou orientadas a objeto.

Redes de Petri

Redes Clássicas

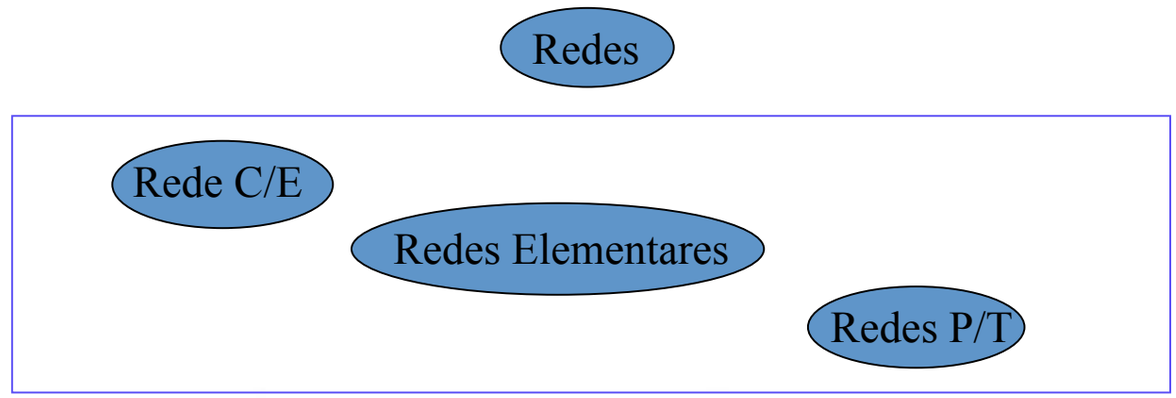
Redes P/T (seriam o padrão)

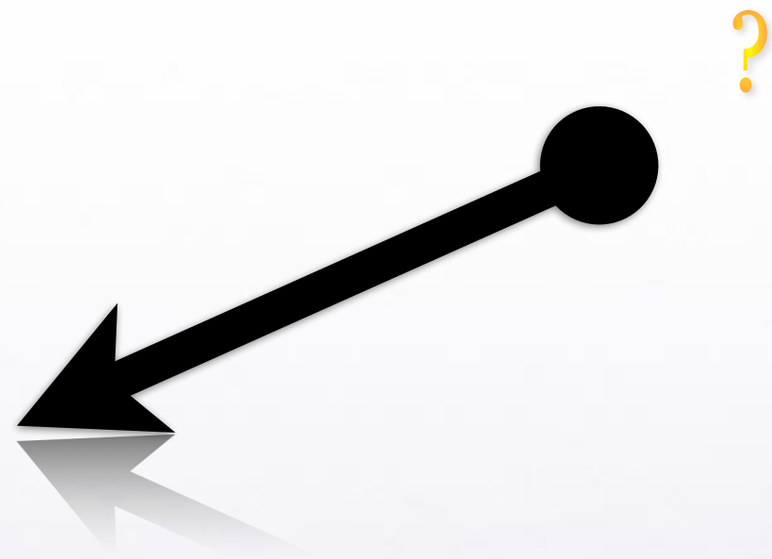
Redes de Alto Nível

(HLPNs, Redes Coloridas, etc)

Redes Estendidas

Redes com elementos estenditos (gates, pseudo-lugares, etc.)
Redes hierárquicas
Redes Orientadas a Objetos
Redes temporais





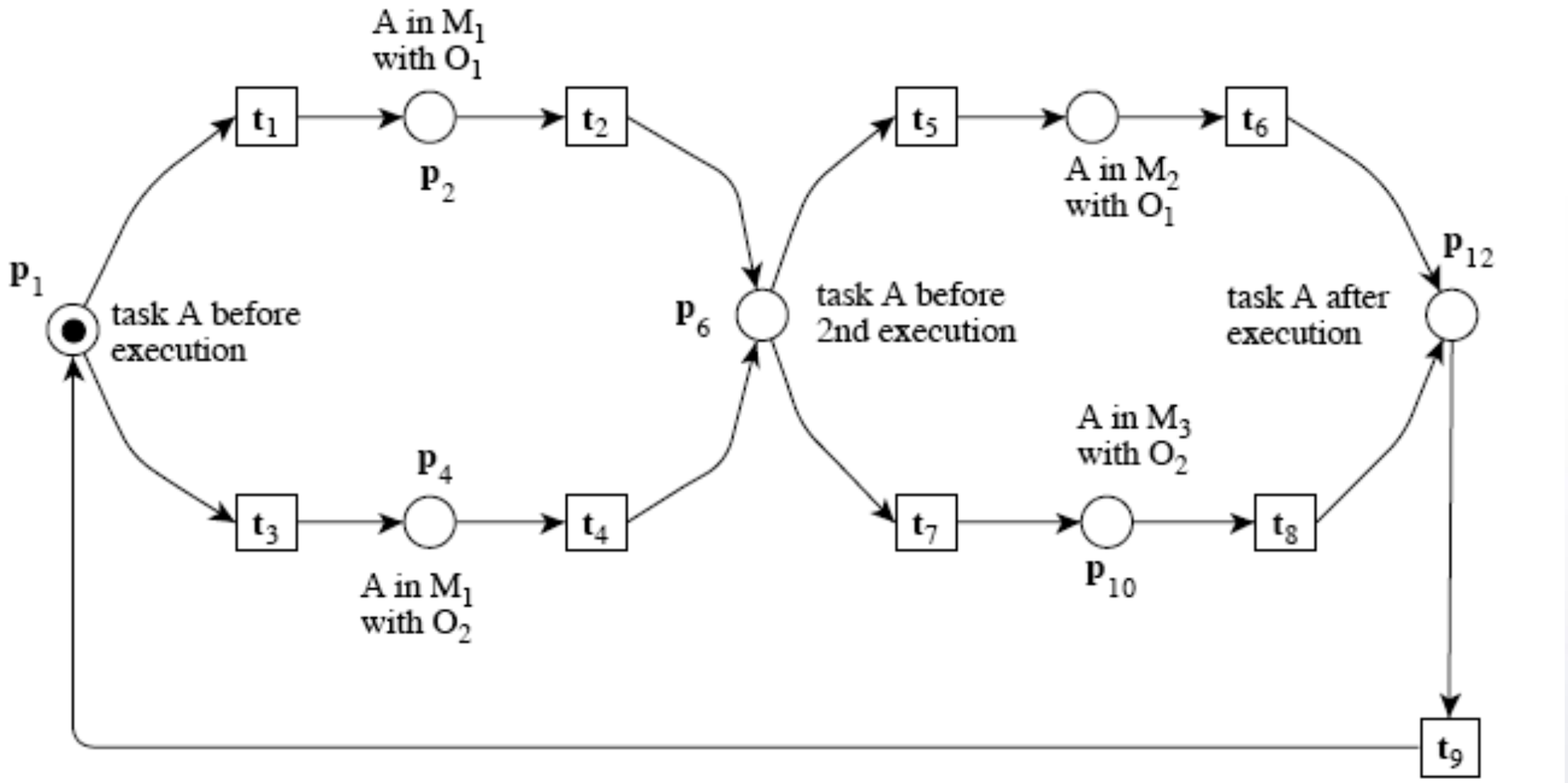
Modelagem de Sistemas Discretos

- Síntese da rede;
- Procedimentos de redução;
- Análise da rede (atingibilidade, deadlock, etc.);
- Simulação.

Procedimento de modelagem e análise

Requisitos do problema: *Seja um sistema de manufatura simples, composto de 3 máquinas DNC: M1, M2 e M3. Estas máquinas podem executar duas operações diferentes, O1 e O2, de modo que O1 pode ser executado nas máquinas M1 e M2 mas não simultaneamente. Similarmente, O2 pode ser executado em M1 e M3 mas não simultaneamente.*

Uma forma de tratar o problema é em primeiro lugar modelar a sequência de operações, sem levar em conta nenhuma restrição e nenhum recurso.



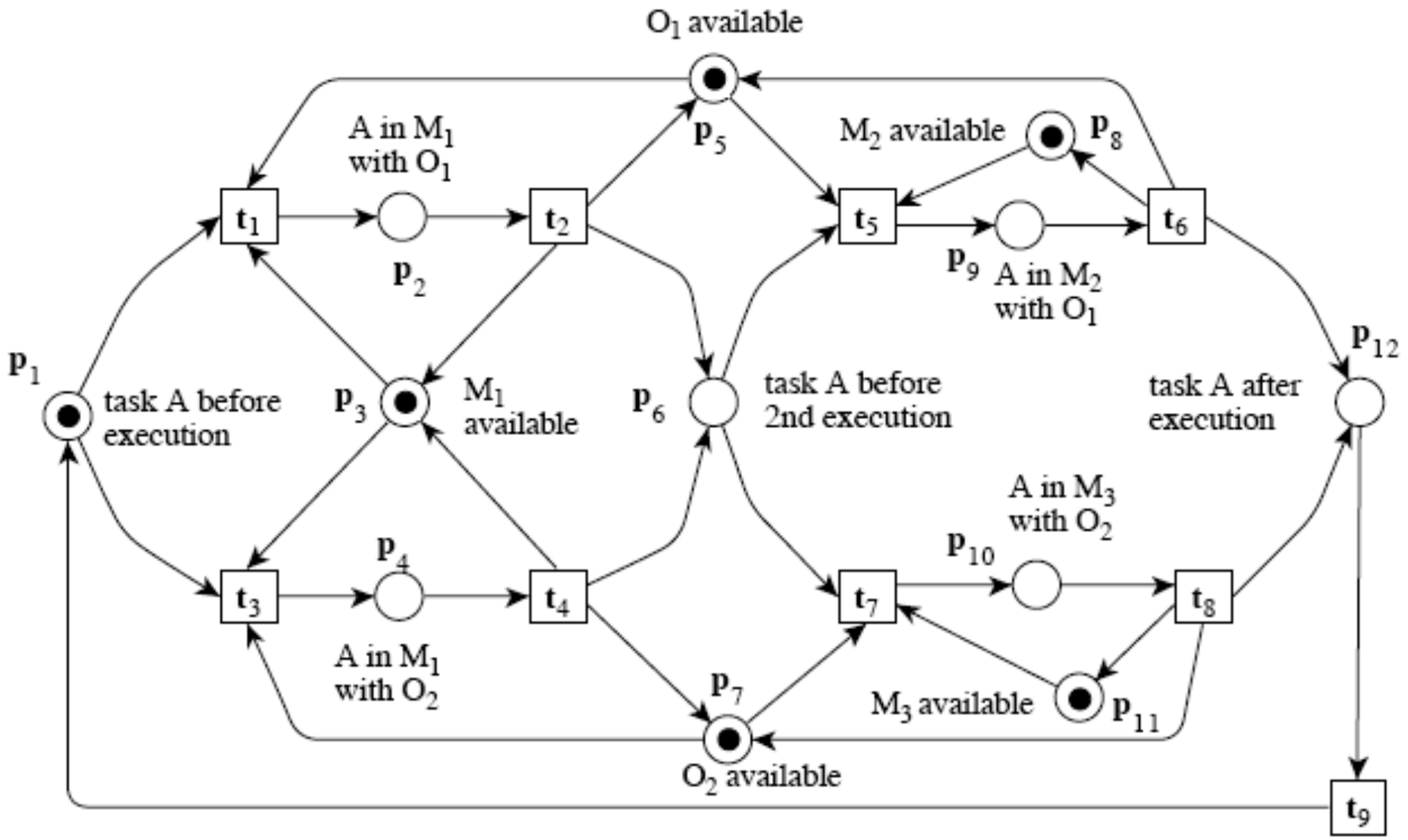
Girault, C. and Valk, R.; Petri Nets for Systems Engineering, Springer-Verlag, 2003

Análise

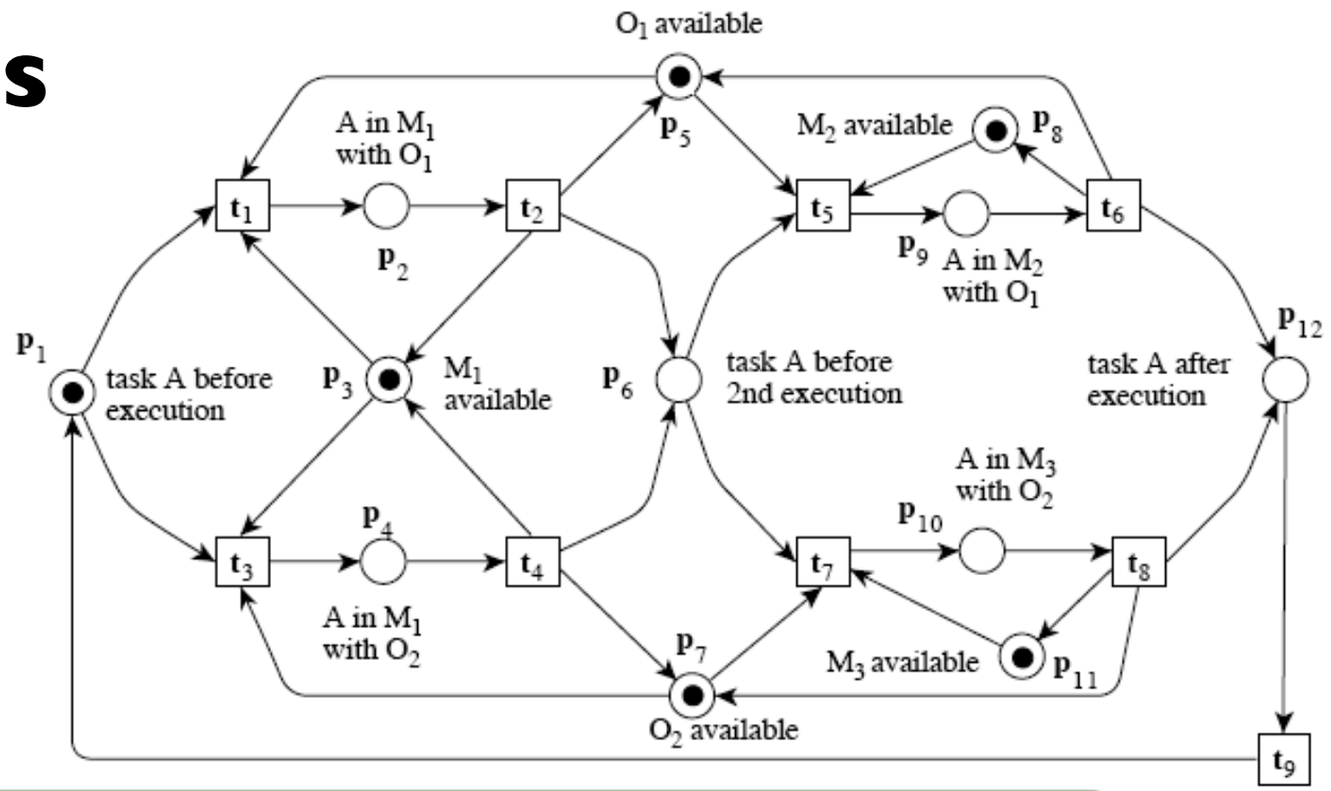
O sistema é cíclico, e permite a combinação das operações em qualquer ordem (e portanto o sistema estaria preparado para implementar qualquer receita de peça). O sistema é conservativo, de modo que cada peça seria representada por uma marca que deve estar em algum dos lugares já especificados.

Na especificação de requisitos, os recursos são representados pela disponibilidade das máquinas e pela sua capacidade de executar cada uma das operações. Uma vez feita a parte sequencial da rede devemos agora introduzir estas restrições, que devem alterar a rede ou a sucessão de estados desta.

Introduzindo os recursos, segundo a especificação de requisitos já apresentada, temos:



Análise de Invariantes



Análise de invariantes

Os invariantes podem ser analisados e servir como forma de verificação para o atendimento dos requisistos

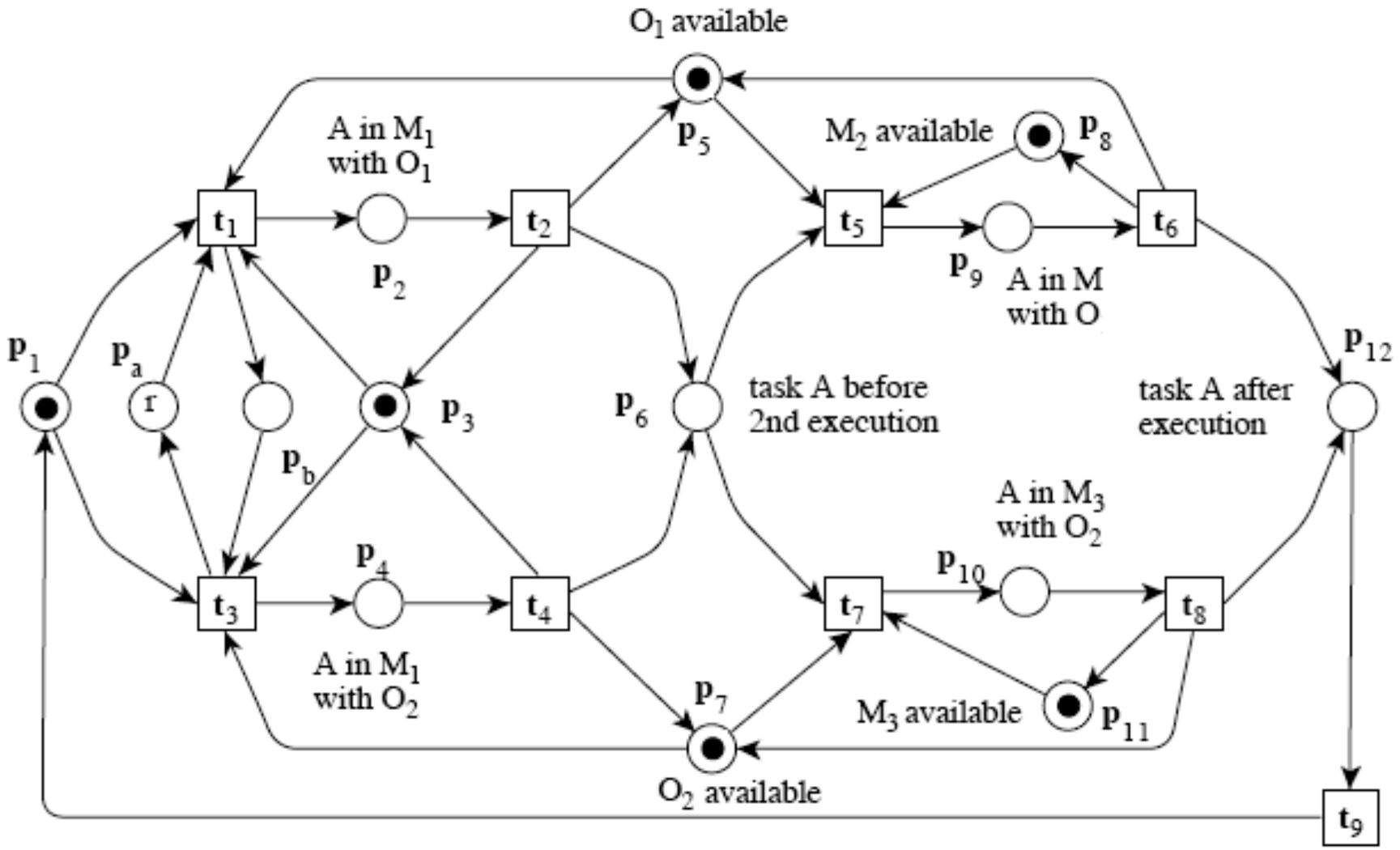
i) $m[p_2] + m[p_5] + m[p_9] = 1;$

ii) $m[p_4] + m[p_7] + m[p_{10}] = 1;$

iii) $m[p_2] + m[p_3] + m[p_4] = 1;$

iv) $m[p_1] + m[p_2] + m[p_4] + m[p_6] + m[p_9] + m[p_{10}] + m[p_{12}] = c$

Introduzindo Sincronização

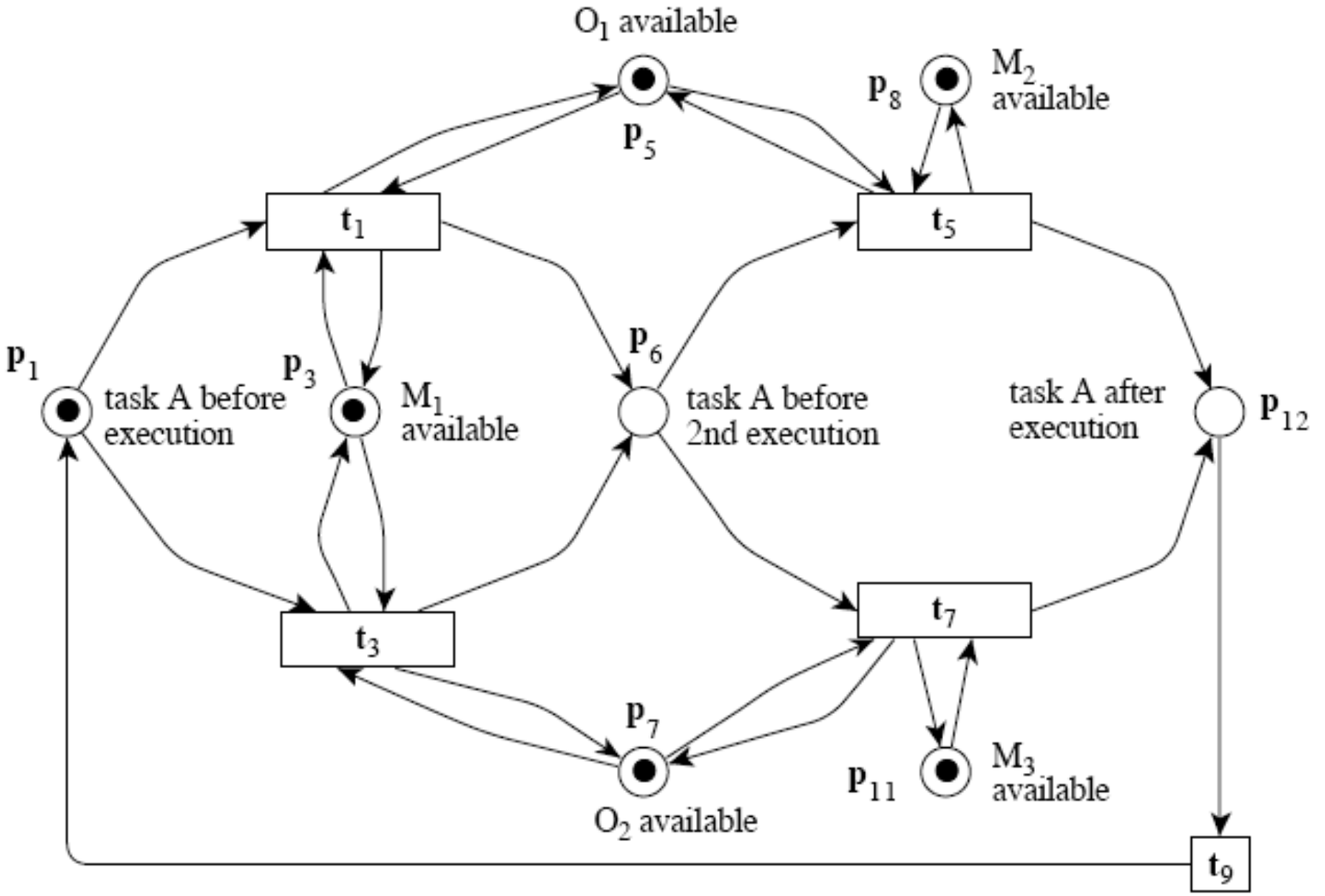


Distância Síncrona

Definition 45

Define-se como a distância síncrona entre duas transições t_1e t_2 de uma rede P/T (N, M_0), ao número inteiro,
$d_{1,2} = max|\overline{\sigma}(t_1) - \overline{\sigma}(t_2)|$,
onde $\overline{\sigma}(t_i)$ é a variância no número de disparos de t_i.

Reduções


O_1 available
p_5
p_8
M_2 available
t_1
t_5
p_1
task A before execution
p_3
M_1 available
p_6
task A before 2nd execution
task A after execution
p_{12}
t_3
t_7
p_7
O_2 available
p_{11}
M_3 available
t_9

Buscando um processo de projeto

Nos casos em que intuitivamente temos um sistema que é plenamente representado por uma rede clássica, e, mais do que isso, onde este modelo é facilmente e completamente interpretado modelo, é fácil de entender que somente uma demanda por múltiplos casos de simetria ou dobramento nos levaria a apelar para um sistema de alto nível.

No exercício que acabamos de ver poderíamos introduzir vários tipos de peça no processo de fabricação, cuja "receita" seria dada por diferentes combinações das operações utilizadas operando nas diferentes máquinas. Neste caso seria bastante atraente a distinção de marcas por tipos. Mas será esta a sequencia adequada em todos os casos?

Requisitos: o início de um grande problema

Certamente o início de todo projeto bem sucedido é a *eliciação* de um conjunto de requisitos que descreve com precisão as funcionalidades do sistema que deve ser modelado e implementado. Portanto para se chegar a um processo de projeto que termine na modelagem do sistema em Redes de Petri é preciso ter em conta de que este projeto deve começar com uma boa representação de requisitos. A representação mais usada e difundida para isso é com certeza a UML.

Análise de Requisitos, Síntese de redes, Building blocks

Uma hipótese bastante tentadoras seria ter um processo de projeto que pudesse ser reduzido a uma sequencia de transformações de transferência semântica entre linguagens, começando pela UML. Uma rede de Petri derivada de um ou mais diagramas UML poderia servir de base para um processo de **análise destes requisitos** e mais tarde com as devidas mudanças inseridas resultar no modelo do sistema.

Este processo poderia perfeitamente ser combinado com o método conhecido como ***building blocks*** onde várias partes da rede poderiam ser sintetizadas como descrito no parágrafo acima até se ter, por composição o sistema completo.

Partindo dos Casos de Uso

Casos de uso podem ser representados segundo uma forma diagramática como proposto em (Silva e Santos, 2004)

Symbol	Description
●	Start of a process (Use Case)
⦿	End of a process (Use Case)
→	Start of an event of basic flow of process
↳	Start of an event of alternative flow of process
◇	Start of a conditional event
$\rightsquigarrow^n$	Jump of current iteration to iteration ^{n}
||	Sequence of concurrent events

Silva, J.R. e Santos, E.A.; Applying Petri Nets to Requirements Validation, ABCM Symposium Series in Mechatronics, vol 1, pp. 508-517.

Um exemplo: o caixa eletrônico

Caso de uso textual

1. Initiate Withdraw – Customer inserts bank's card in the card reader on the ATM machine
2. Verify Bank Card – The ATM reads the account code from the magnetic strip on the bank card and checks if it is an acceptable bank card
3. Enter PIN – The ATM asks for the customer's PIN code (4 digits)
4. Verify PIN – The account code and PIN are verified to determine if the account is valid and if the PIN entered is the correct PIN for the account. For this flow, the account is a valid account and the PIN is the correct PIN associated with this account
5. Select Withdraw – The ATM displays the different alternatives available at this ATM. In this flow, the bank customer always selects "Cash Withdraw"
6. Enter Amount – The ATM asks for the amount to withdraw. For this flow the customer selects a pre-set amount ($10, $20, $50, or $100)
7. Authorization – The ATM initiates the verification process with the Banking System by sending the Card ID, PIN, Amount, and Account information as a transaction. For this flow, the Banking System is online and replies with the authorization to complete the cash withdrawal successfully and updates the account balance accordingly
8. Dispense – The Money is dispensed
9. Receipt – The receipt is printed and dispensed. The ATM also updates the internal log accordingly
10. Return Card – The Bank Card is returned

1 ● Initiate Withdraw – Customer inserts bank's card in the card reader on the ATM machine; (

1.1 ◇ Is the card valid? Verify Bank Card The ATM reads the account code from the magnetic strip on the bank's card and checks if it is an acceptable card;

1.1.1 ↳Send message of invalid card – If the card isn't an acceptable card, send an appropriate message. ⇝1.9

1.2 → Enter PIN – The ATM asks for the customer's PIN code (4 digits);
1.3 ◇ Is PIN Correct? – Verify PIN – The account code and PIN are verified to determine if the account is valid and if the PIN entered is the correct PIN for the account;

1.3.1 ◇ Is the account valid? – Verify account code – The account code is verified;

1.3.1.1 ↳Send message of invalid account – The Banking system returns a code indicating the account could not be found or is not an account which allows withdrawals. ⇝1.9

1.3.2 ◇ Is the final try? – Checks the number of tries – The customer has three tries to enter the correct PIN;

1.3.2.1 ↳Send message of incorrect PIN – The ATM send an appropriate message. ⇝1.2

1.3.3 ↳Retain card – on the final try the card is retained, ATM returns to Ready State, and this use case terminates. ⇝2

1.4 ◇ Have money the ATM? – Select Withdraw – The ATM displays the different alternatives available at this ATM. In this flow, the bank customer always selects "Cash Withdraw";

1.4.1 ↳Send message ATM out of Money – If the ATM is out of money, the "Cash Withdraw" option will not be available. ⇝1.4

1.5 ◇ Sufficient funds and does not exceed the daily limit? – Enter amount – The ATM asks for the amount to withdraw. For this flow the customer selects a pre-set amount ($10, $20, $50, or $100);

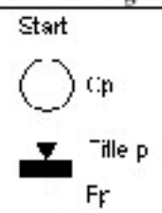
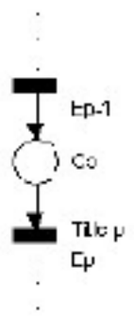
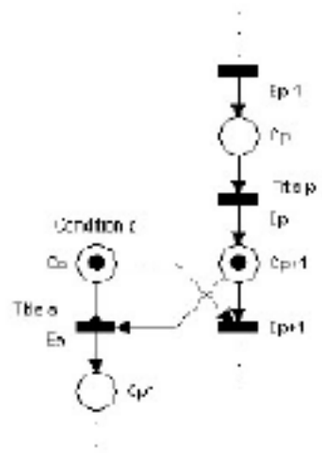
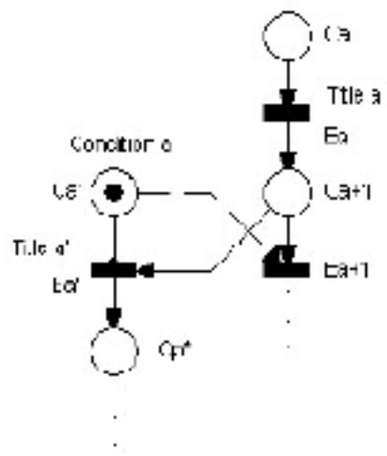
1.5.1 ◇ Sufficient funds? – Check sufficient funds – Check if the funds are sufficient;

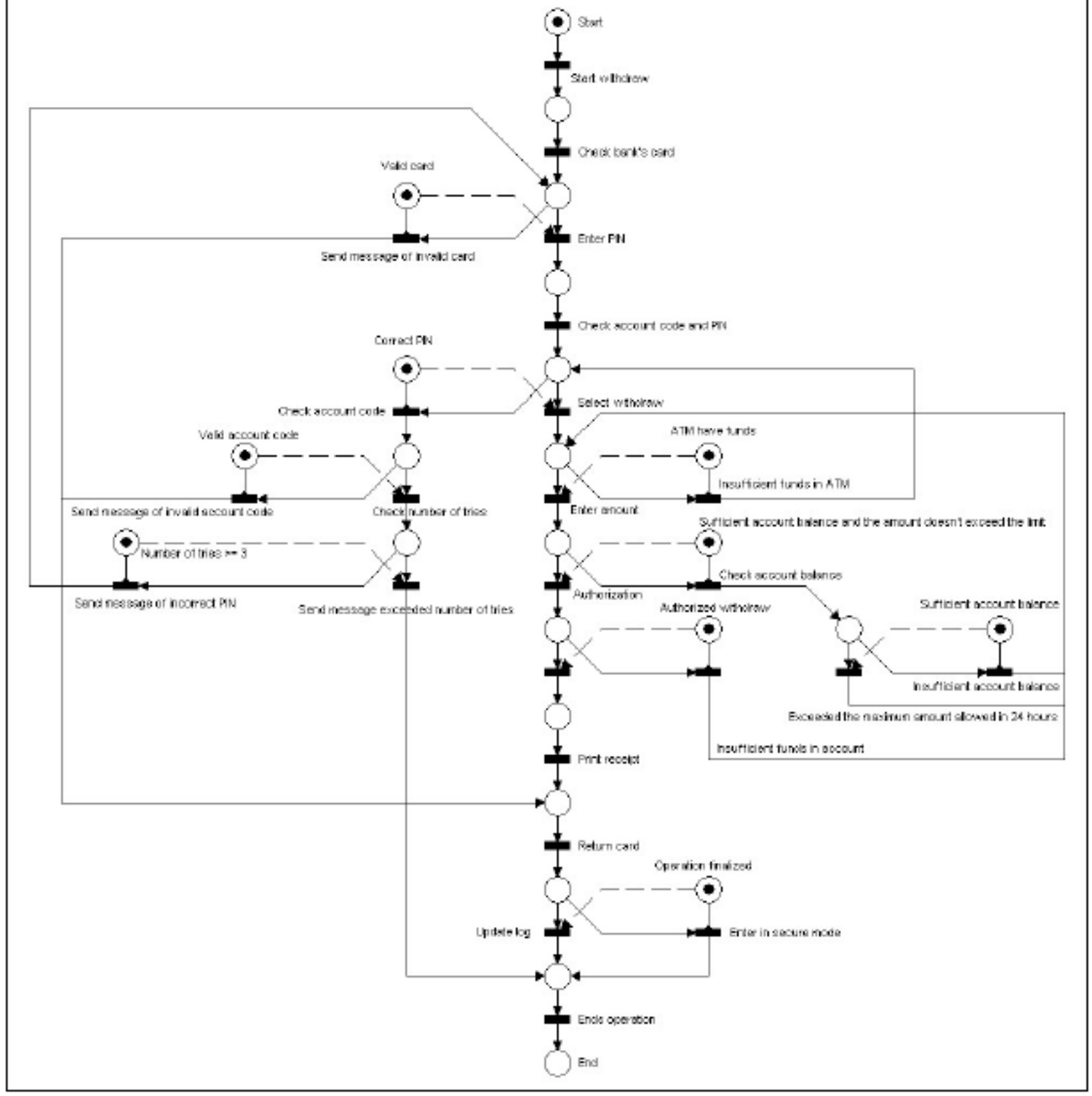
1.5.1.1 ↳Send message insufficient funds in ATM – The ATM contain insufficient funds to dispense the requested amount. ⇝1.5

```
<BF>          ::= <initial> { <eventB> } <final> { <AF> }
<initial>     ::= <label> "●" <title> " " <description> ";" "("
<label>       ::= <number> { "." <number> }
<number>      ::= <sequential number>
<title>       ::= <string>
<description> ::= <string>
<string>      ::  <any_caractor> { <any_caractor> }
<eventB>      ::= [ <eventBB> | <eventBA> ]
<eventB>      ::= <labelB> [ ( "→" <title> " " <description> ";" ) |

                             ( "◇" <condition> "-" <title> "-" <description> ";" <eventA>
                             ) ] { <eventB> } |

                  "||" "(" <eventB> { <eventB> } ")" "(" <eventB> { <eventB> } ")" "||" {
                  <eventB> }
<labelB>      ::= <label> "." <number>
<condition>   ::= <string>
<eventA>      ::= <labelA> [ ( "↳" <title> " " <description> <branch> ) |

                             ( "◇" <condition> "-" <title> "-" <description> ";" <eventA>
                             ) ] { <eventA> }

<labelA>      ::= <labelB> "." <number>
<branch>      ::= "↷" [ <label> | <labelB> ]
<final>       ::= ")" <label> "◉" <title> "-" <description> ";"
<AF>          ::= <eventUA>
<eventUA>     ::= <label> "|" <title> "-" <description> <branch> { <eventUA> }
```

Where:

- eventB → Event of Basic flow
- labelB → Event id of Basic flow
- eventA → Alternative Event of Basic flow
- labelA → Alternative Event id of Basic flow
- eventUA → Unconditional Event of Alternative Event
- labelUA → Unconditional Event id of Alternative Event

Event	BNF notation	Petri net segment
Initial	p ● Title - Description;	
Basic	p → Title - Description;	
Basic conditional and alternative normal[1]	p◯Condition-Title-Descript.; a → Title-Description ~P*	
Alternative conditional and normal	a◇Condition-Title-Descript.; a' → Title-Description ~P*	


Start
Start withdraw
Check bank's card
Valid card
Send message of invalid card
Enter PIN
Check account code and PIN
Correct PIN
Check account code
Select withdraw
Valid account code
ATM have funds
Send message of invalid account code
Check number of tries
Insufficient funds in ATM
Enter amount
Number of tries == 3
Sufficient account balance and the amount doesn't exceed the limit
Check account balance
Send message of incorrect PIN
Send message exceeded number of tries
Authorization
Authorized withdraw
Sufficient account balance
Insufficient account balance
Exceeded the maximum amount allowed in 24 hours
Insufficient funds in account
Print receipt
Return card
Operation finalized
Update log
Enter in secure mode
Ends operation
End

Andamento do curso

Meta -> capítulo 9 do livro texto

Leitura da semana

Capítulos 5 e 6 (revisão)

Capítulos 7 e 8 (modelagem)

Milestone 5 -> feedback

Milestone 6 (primeira versão do artigo completo)

Teremos mais duas semanas de aula

Fim