# *PMR 5237*

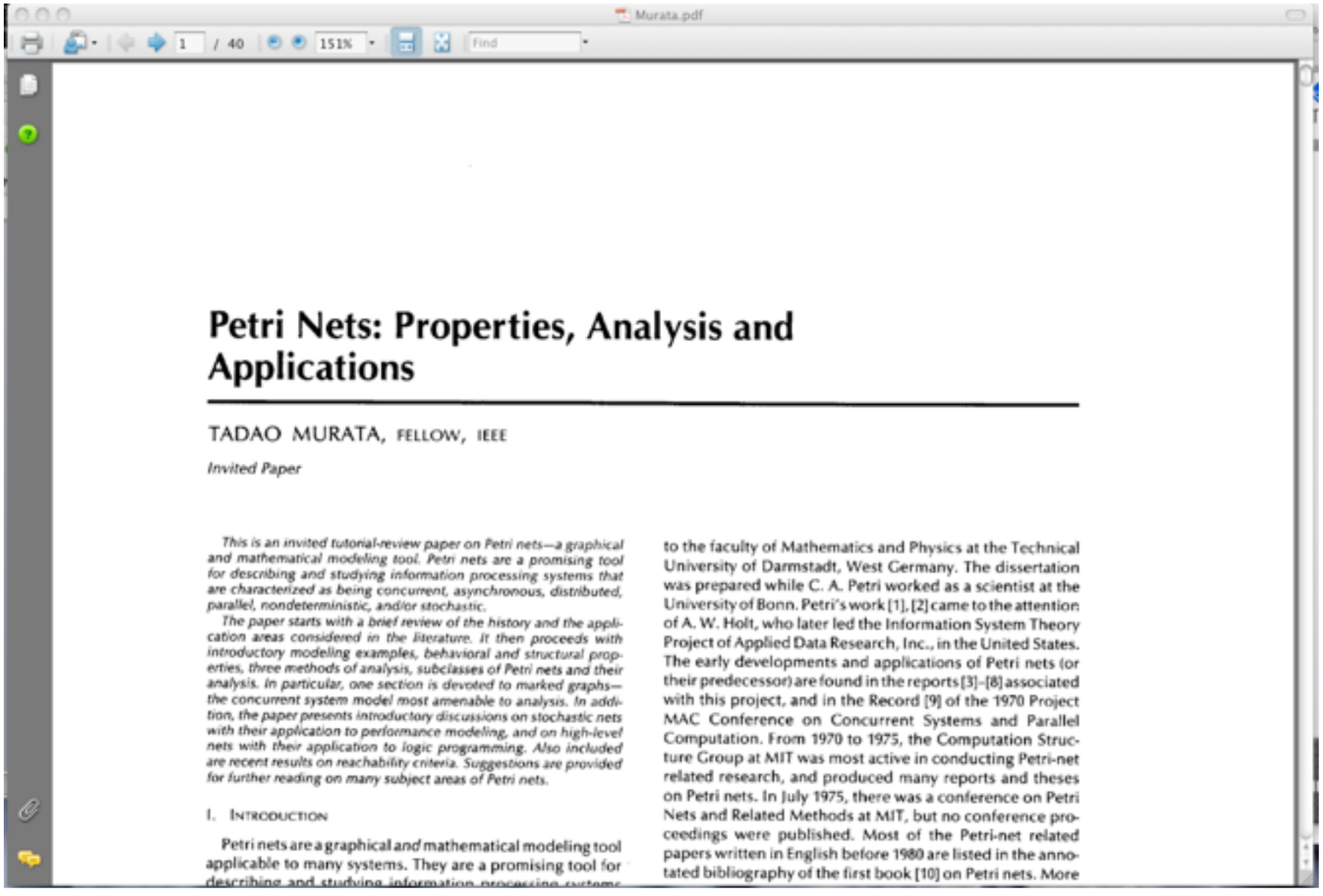
# Modelagem e Design de Sistemas Discretos em Redes de Petri

## Aula 6: Redes de Alto Nível

Prof. José Reinaldo Silva
reinaldo@usp.br

# Leitura da Semana

Tadao Murata



Redes

Rede C/E

Redes Elementares

Redes P/T

?

IEC/ISO 15909

Parte 1 (2004): modelo semântico, definição teórica das redes clássicas e das **redes de alto nível.**

Parte 2 (2005-2008): definição do protocolo de importação/exportação, PNML.

Parte 3 ( ? ) : Extensões, Redes Temporizadas, modularidade, hierarquia.

# A dialética do processo de design

- Relações de vizinhança
- Negação das relações e restrições

# Usando redes de Petri para análise de Requisitos (Use-Case)

**Verification of Use Case with Petri Nets in Requirement Analysis***

Jinqiang Zhao[1,2] and Zhenhua Duan[1,**]

[1] Institute of Computing Theory & Technology, Xidian University, Xi'an, 710071, P.R. China
[2] State Key Laboratory of Software Engineering, Wuhan University, 430072, P.R. China
jqzhao1985@gmail.com, zhenhua_duan@126.com

**Abstract.** Requirement analysis plays a very important role in reliability, cost, and safety of a software system. The use case approach remains the dominant approach during requirement elicitation in industry. Unfortunately, the use case approach suffers from several shortcomings, such as lacking accuracy and being difficult to analyze and validate the dynamic behavior of use cases for concurrency, consistency, etc. This paper proposes an approach for overcoming limitations of the use case approach and applies the approach in Model Driven Development (MDD). Timed and Controlled Petri Nets are used as the formal description and verification mechanism for the acquired requirements. Use cases are used to elicit the requirements and to construct scenarios. After specifying the scenarios, each of them can be transformed into its correspondent Petri-net model. Through analyzing these Petri-net models, some flaws or errors of requirements can be detected. The proposed approach is demonstrated by an E-mail client system.

**Keywords:** use case; Model Driven Development; Petri net; requirement analysis.

## 1 Introduction and Related Works

Software development usually consists of the following stages: requirement analysis, design, code and testing. Many research studies have shown the considerable influence of early requirement analysis on the reduction of the unnecessary costs, confusion and complexity in the later phases of software development [1].

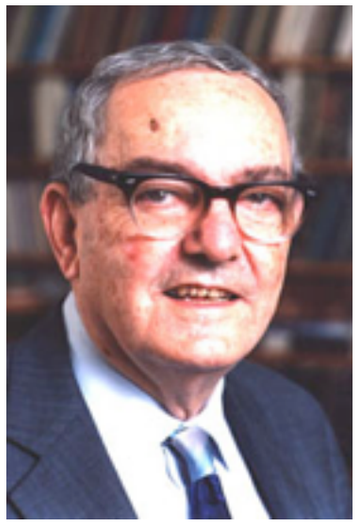
Therefore, high quality of requirement analysis can most likely reduce some potential risk occurred in later phases of software development.

* This research is supported by the NSFC Grant No. 60433010, NSFC Grant No. 60873018 jointly sponsored by Microsoft Asia Research Academy, Defence Pre-Research Project of China No. 51315050105, SRFDP Grant 200807010012 and SKLSE20080713.
** Corresponding author.

O. Gervasi et al. (Eds.): ICCSA 2009, Part II, LNCS 5593, pp. 29–42, 2009.

# O processo de projeto

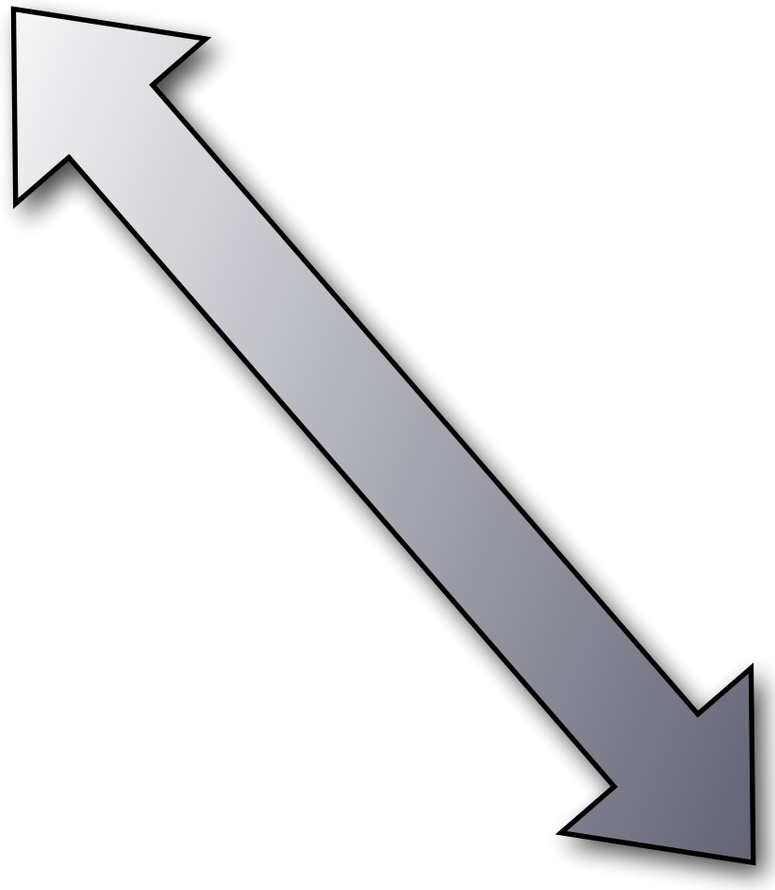
| Ciclo de vida clássico | Características | Possibilidade do uso de Redes de Petri |
|---|---|---|
| Requisitos | Informal, abstrata | Pesquisa em aberto |
| Especificação | Formal | Redes clássicas e de alto nível |
| Modelagem e Design | Método de espaço de estados<br>Modelagem da parte estática e da parte dinâmica | Redes clássicas e de alto nível |
| Construção | Execução, síntese automática de programas executáveis | Redes clássicas |



Herbert Alexander Simon
1916-2001
Turing Award, 1975
Nobel Prize in Economy, 1978
US National Medal of Science, 1986
Psychology Award, 1993

# abstrações

modelos



# implementação

processos

# Applications

Industry Applications

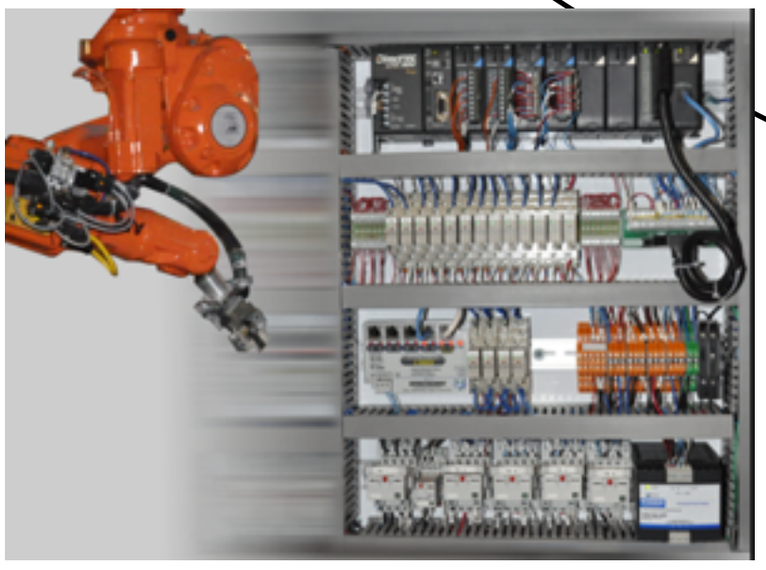
manufacturing

*PLC's*

process industry

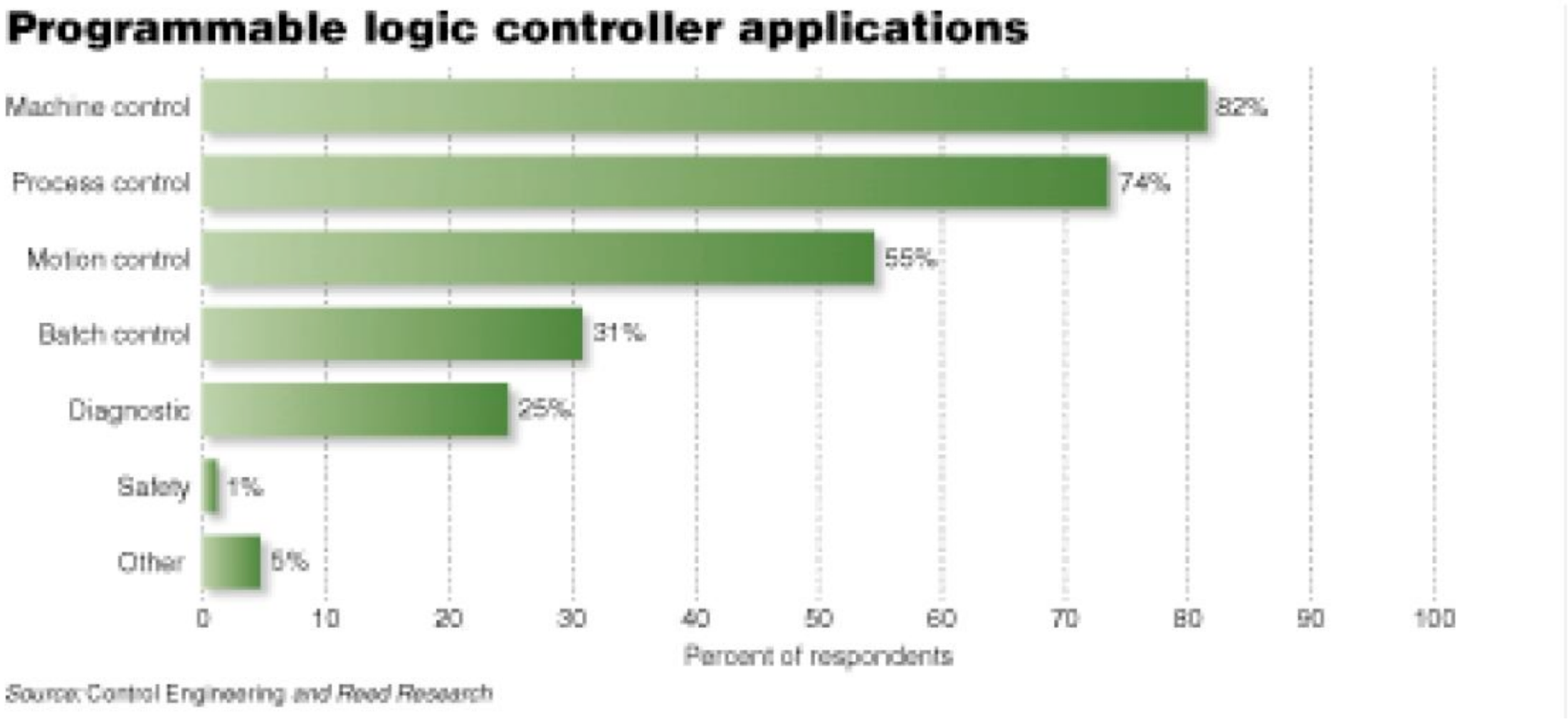
*SDCD's*

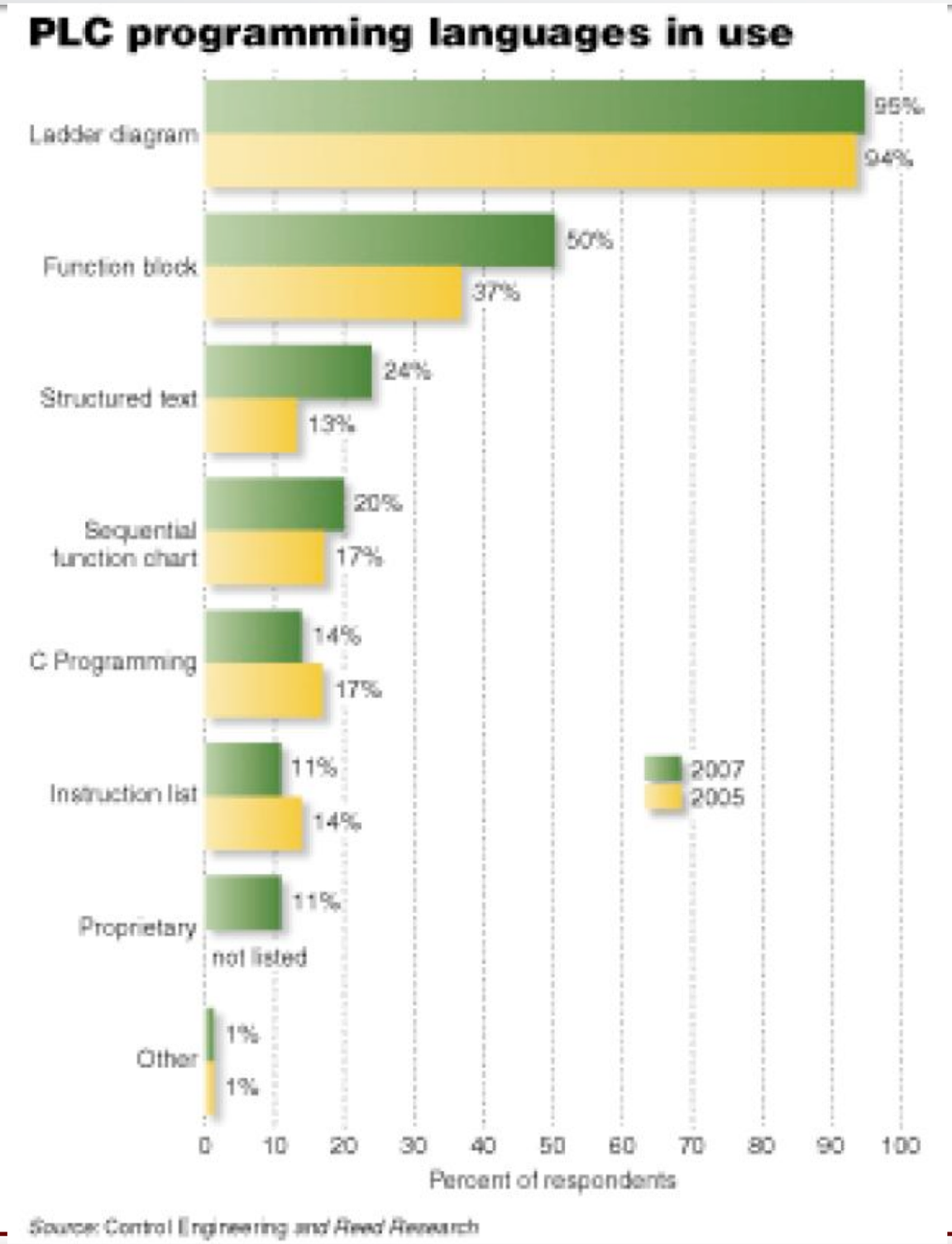
(Abstract) Models



# Aplicações das RdP em processos industriais

*Dick Johnson, Control Engineering -- Control Engineering, 12/1/2007*




PLC programming languages in use
Ladder diagram
95%
94%
Function block
50%
37%
Structured text
24%
13%
Sequential function chart
20%
17%
C Programming
14%
17%
Instruction list
11%
14%
Proprietary
11%
not listed
Other
1%
1%
2007
2005
0
10
20
30
40
50
60
70
80
90
100
Percent of respondents
Source: Control Engineering and Reed Research

O problema entre modelagem (modelo) e a sua implementação é que durante a modelagem algumas propriedades "abstratas" são mais visíveis, ou visíveis somente neste nível de abstração. Na implementação outras propriedades são mais salientes.

Por exemplo, as propriedades de simetria, os invariantes, e propriedades do modelo que discutimos em aula são visíveis na modelagem. Vivacidade, presença de sifões ou tracks são visíveis na implementação ou simulação.

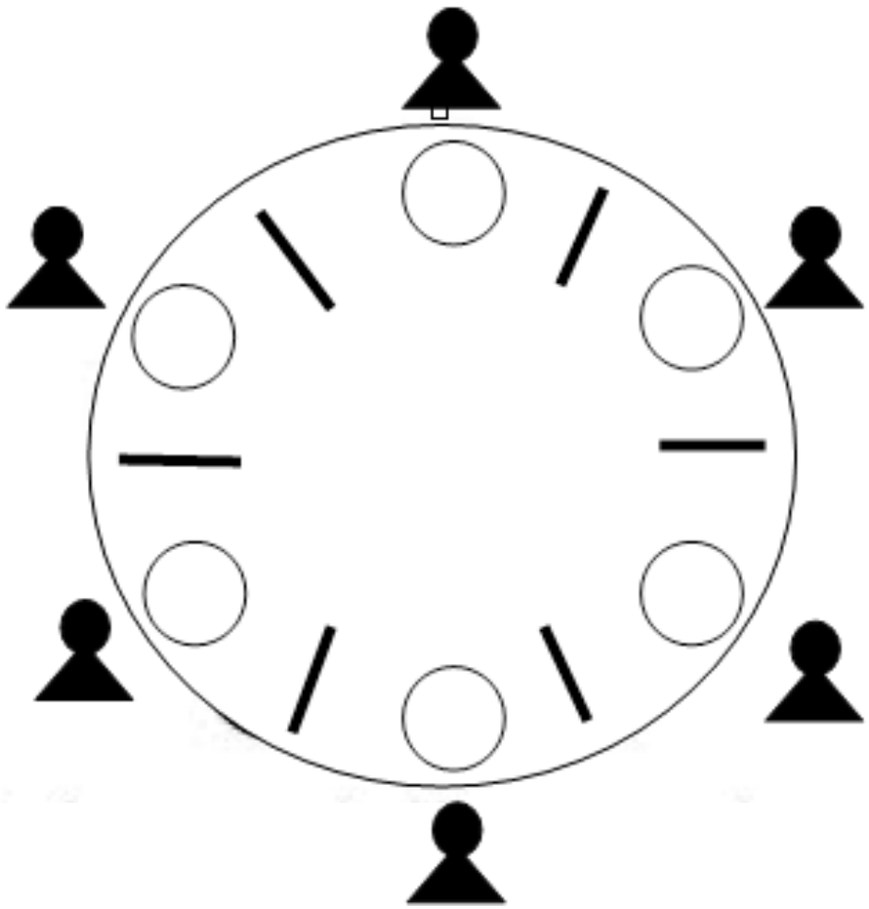
Existem propriedades que são mais visíveis e mensuráveis (e úteis) somente quando se tem uma visão mais abstrata dos artefatos. Por exemplo a *simetria*.

> The term “symmetry” derives from the Greek words *sun* (meaning ‘with’ or ‘together’) and *metron* (‘measure’), yielding *summetria*, and originally indicated a relation of commensurability (such is the meaning codified in Euclid's *Elements* for example). It quickly acquired a further, more general, meaning: that of a proportion relation, grounded on (integer) numbers, and with the function of harmonizing the *different* elements into a *unitary whole*. From the outset, then, symmetry was closely related to harmony, beauty, and unity, and this was to prove decisive for its role in theories of nature.

Stanford Encyclopedia of Philosophy

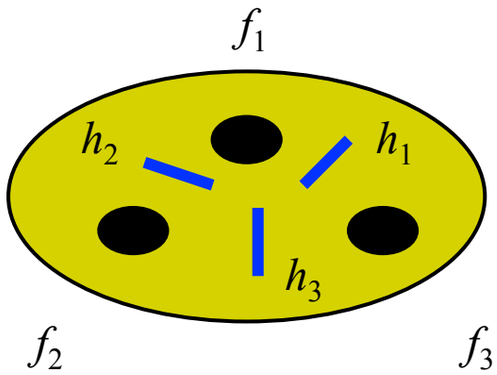
# The dinning philosophers problem

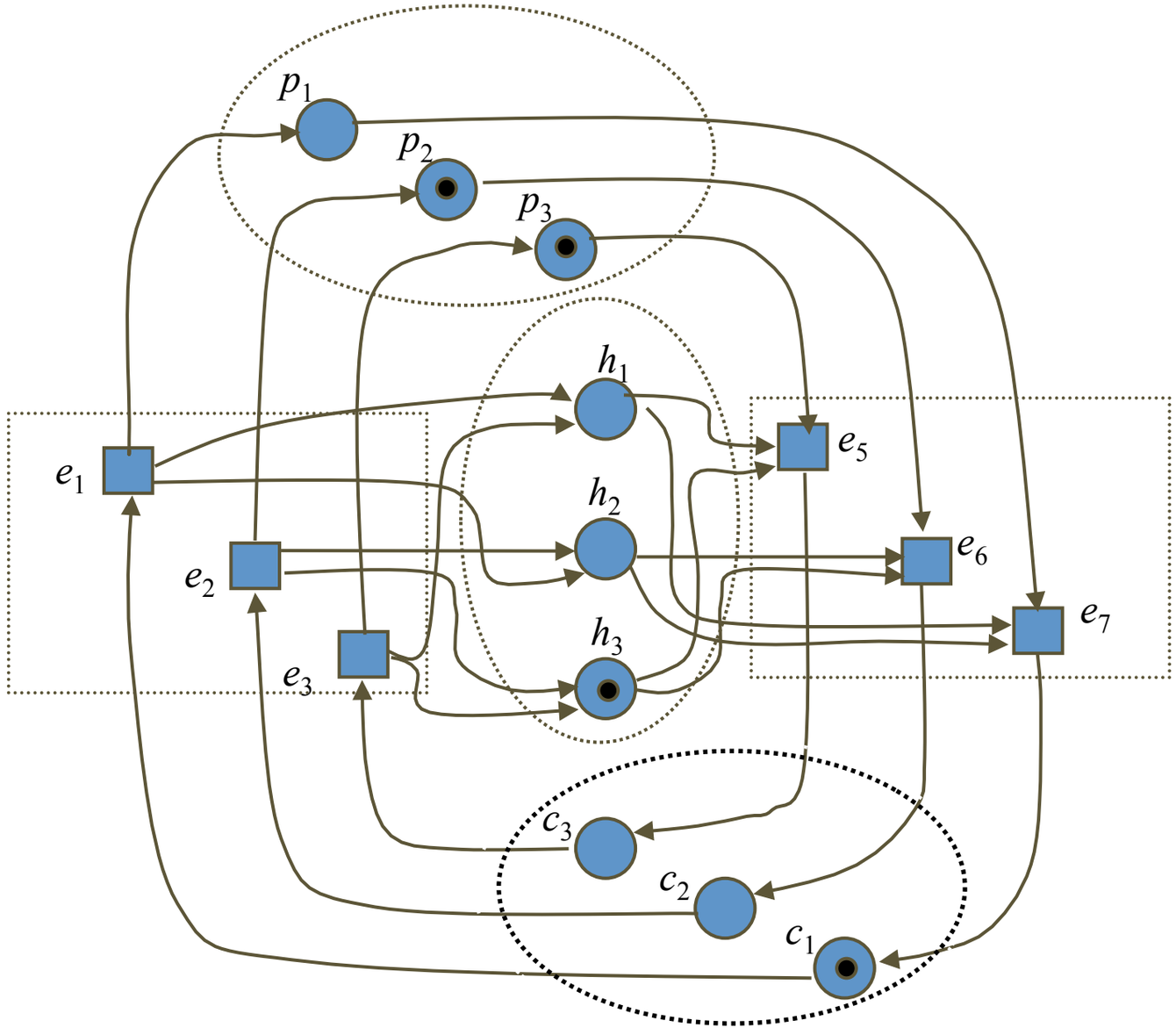
Cinco filósofos sentam em uma mesa e podem alternadamente pensar ou comer. No entanto, para esta última tarefa devem usar os utensílios ao lado do prato (hashis) o que impede os seus vizinhos de fazer a mesma coisa. O problema original prevê cinco filósofos sendo que somente dois deles conseguem passar do estado pensao para comendo simultaneamente.



# Dobramento em RdP

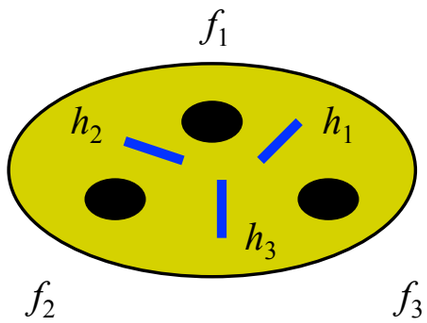
## O exemplo dos filósofos

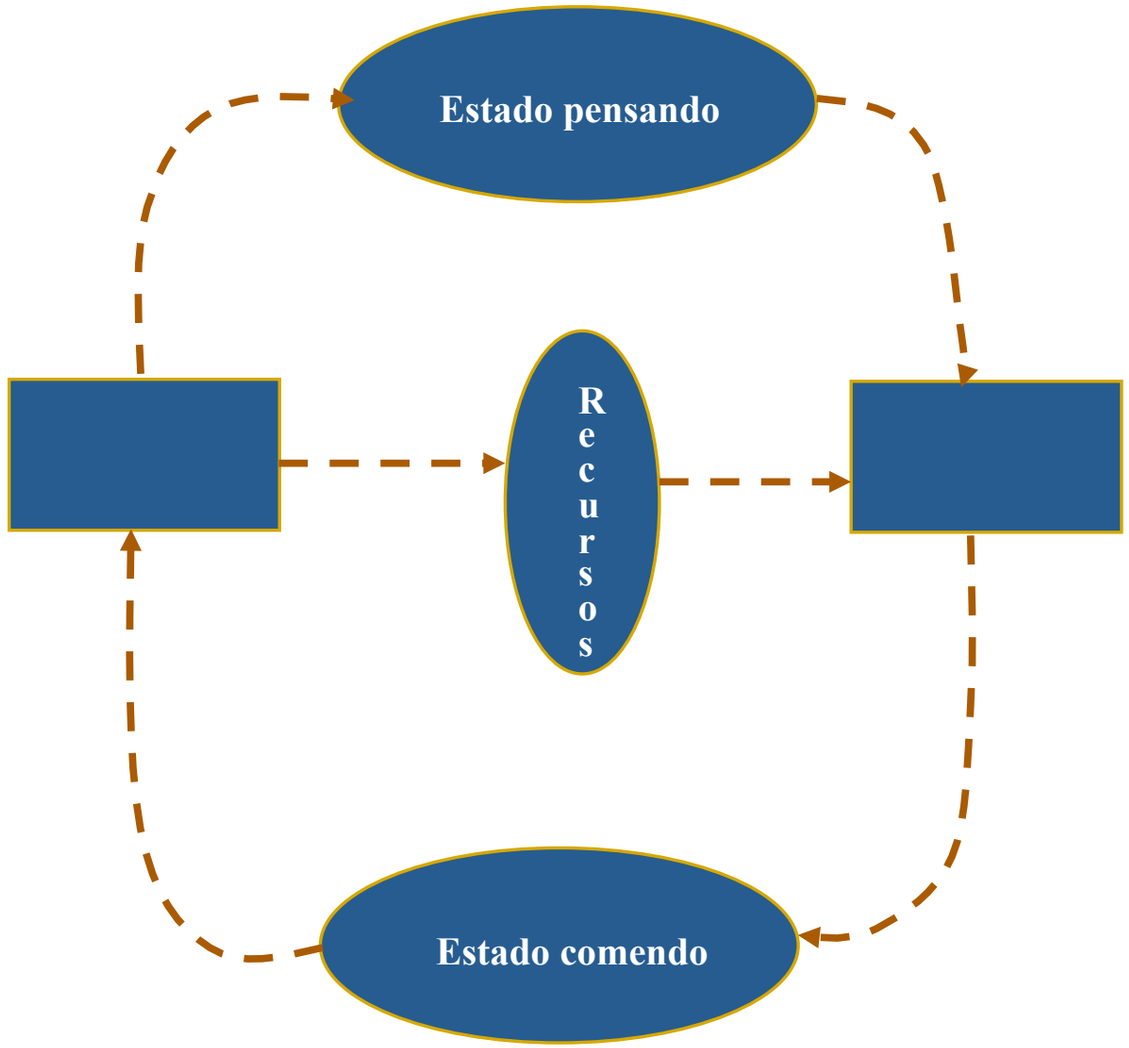


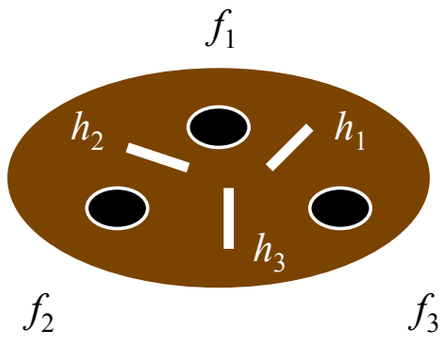
# O dobramento

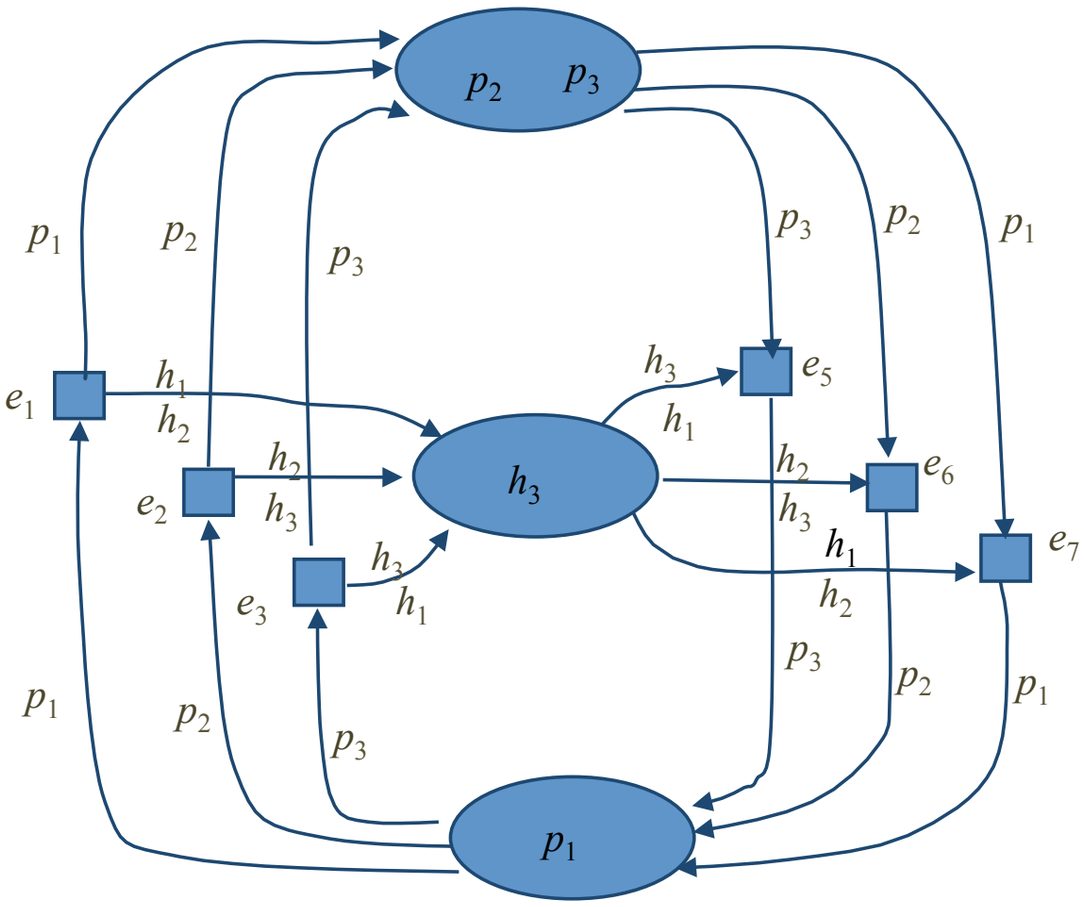
O exemplo dos filósofos



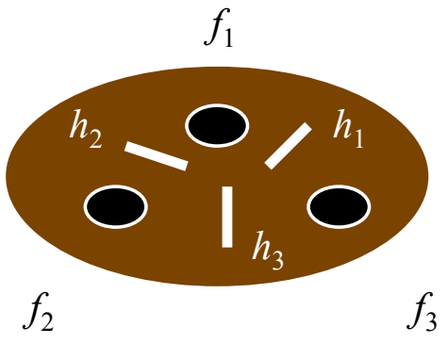
# O exemplo dos filósofos



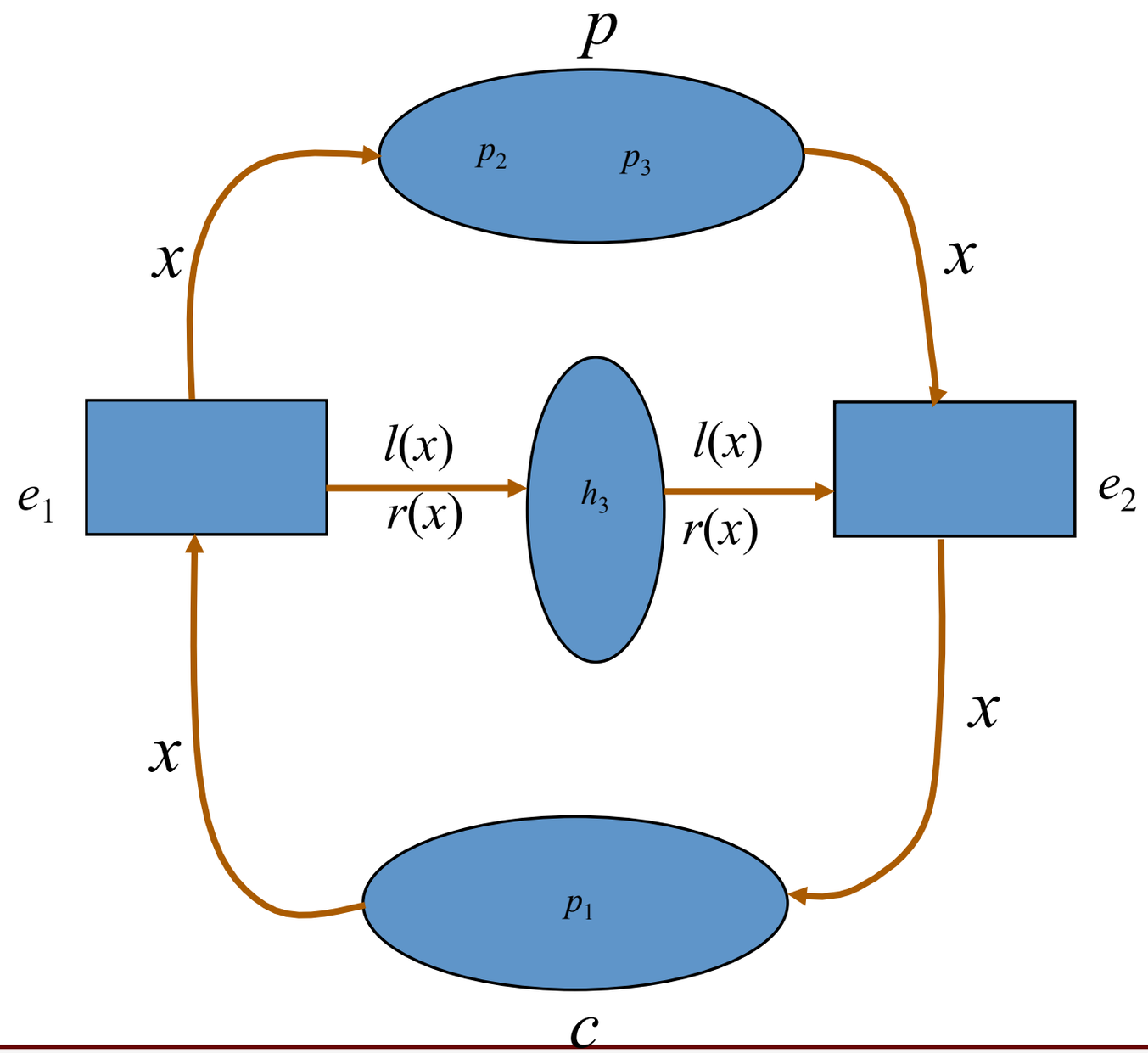

O colapso dos lugares leva à necessidade de se distinguir as marcas, tanto as que representam os filósofos quanto as que representam os recursos, isto é, os hashis.

# O dobramento completo




$P=\{p_1, p_2, p_3\}$
$H=\{h_1, h_2, h_3\}$
$U=P \cup H$

$l : P \rightarrow H$
$p_i \rightarrow h_i$
$r : P \rightarrow H$
$p_1 \rightarrow h_2$
$p_2 \rightarrow h_3$
$p_3 \rightarrow h_1$

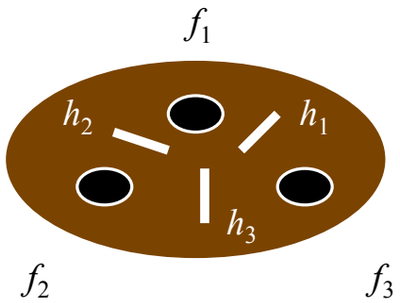


Exercicio 3, cap 8. Reisig

Caracterize conflito na rede PrT que representa o problema dos filósofos. Suplemente esta rede de modo que os três filósofos comam e pensem com igual frequencia, isto é, torne a rede *fair* no que diz respeito aos eventos que fazem o filósofo *i* passar do estado de reflexão ao estado de alimentação.

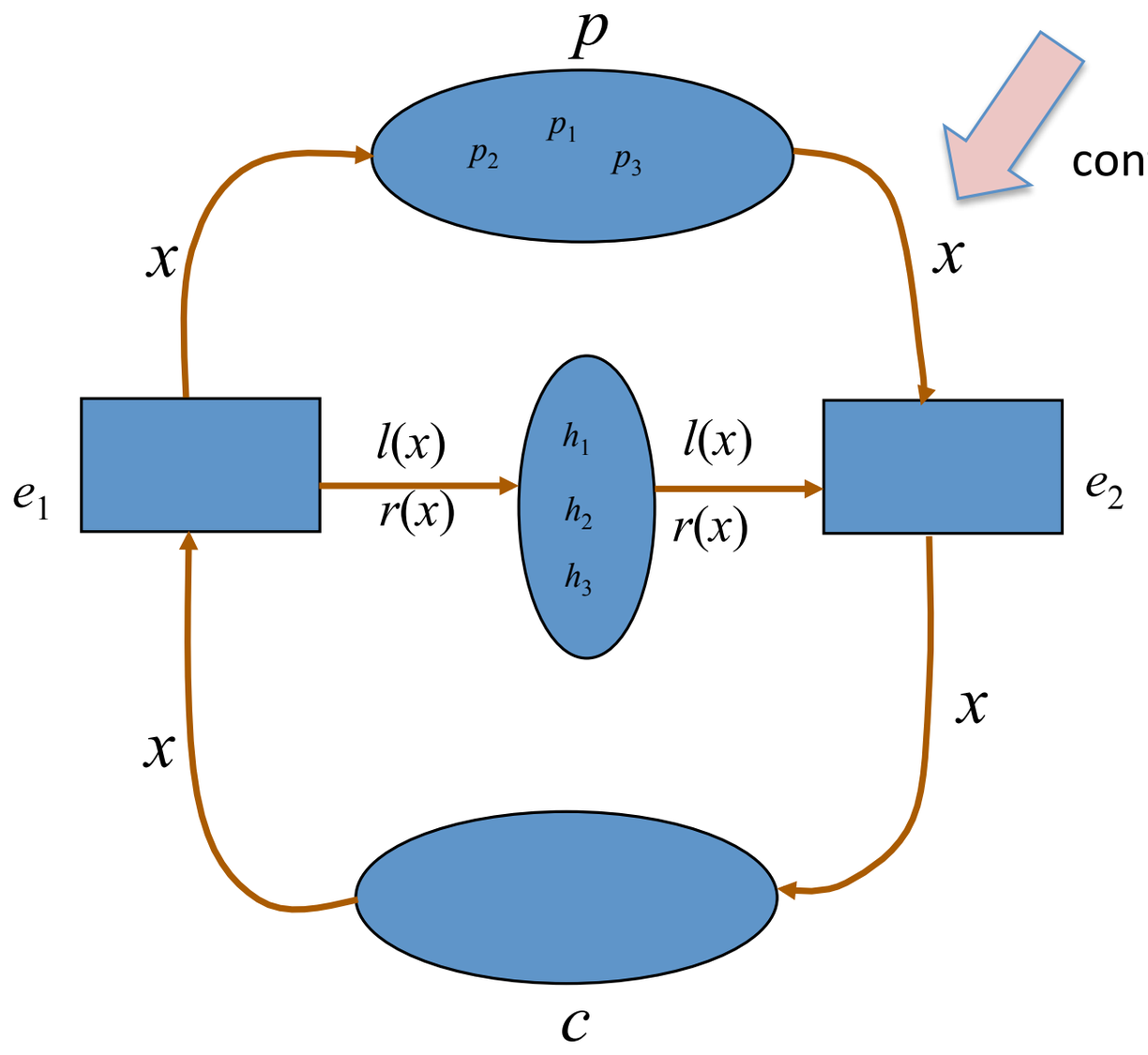
Wolfgang Reisig, Petri Nets: an Introduction, EACTS, Monographs in Theoretical Computer Science, Springer Verlag, 1985

# O exemplo dos filósofos




P=$\{p_1, p_2, p_3\}$
H=$\{h_1, h_2, h_3\}$
U=P ∪ H

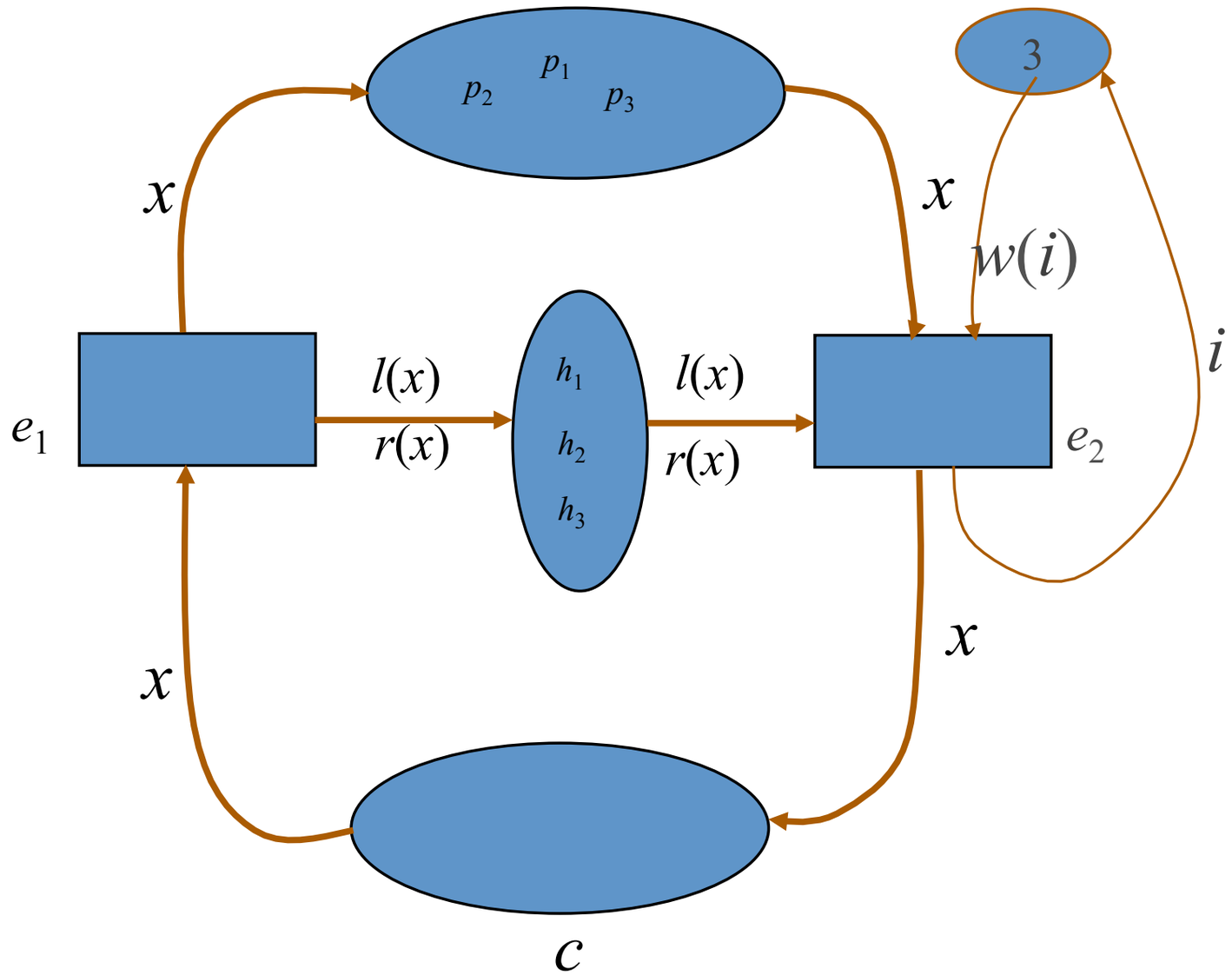
$l$ : P → H
$p_i \rightarrow h_i$
$r$ : P → H
$p_1 \rightarrow h_2$
$p_2 \rightarrow h_3$
$p_3 \rightarrow h_1$



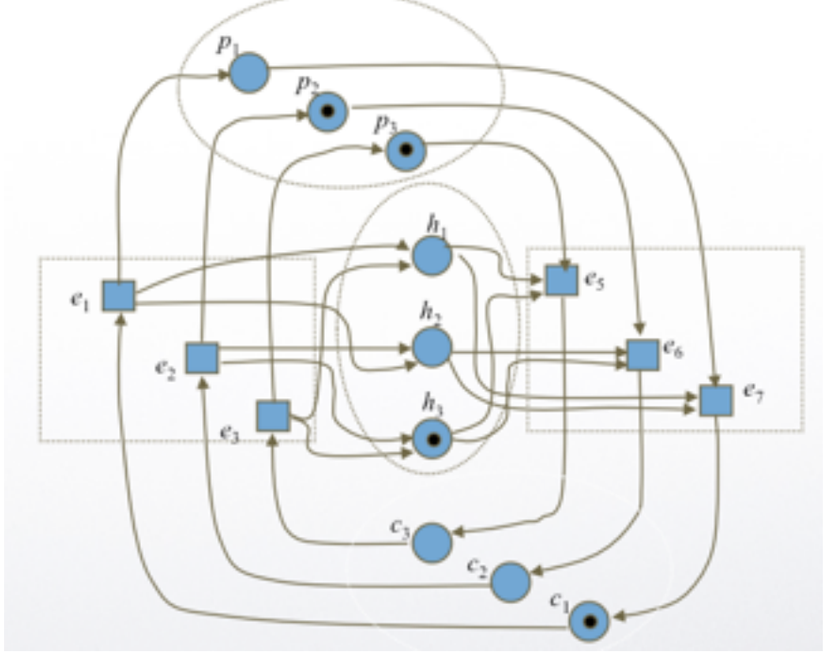
# O exemplo dos filósofos

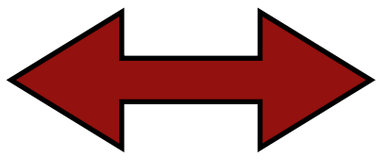
P=$\{p_1, p_2, p_3\}$
H=$\{h_1, h_2, h_3\}$
U=P ∪ H

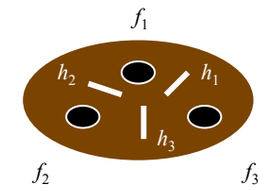
$l$ : P → H
$p_i \rightarrow h_i$
$r$ : P → H
$p_1 \rightarrow h_2$
$p_2 \rightarrow h_3$
$p_3 \rightarrow h_1$
$w$: P → I
$p_j \rightarrow$ j=($i$)mod 3+1



O dobramento leva de fato a uma rede mais compacta com um número menor de lugares e de transições. Entretanto, lembre que esta nova rede NÃO PODE ser separada das inscrições e tipos que a acompanham.

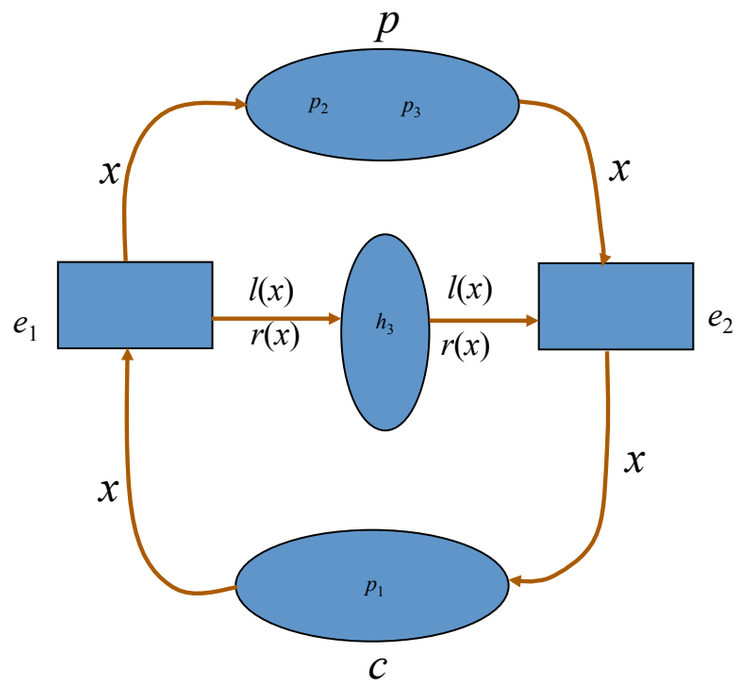





P={$p_1$, $p_2$, $p_3$}
H={$h_1$, $h_2$, $h_3$}
U=P ∪ H

$l$ : P → H
$p_i$ → $h_i$
$r$ : P → H
$p_1$ → $h_2$
$p_2$ → $h_3$
$p_3$ → $h_1$



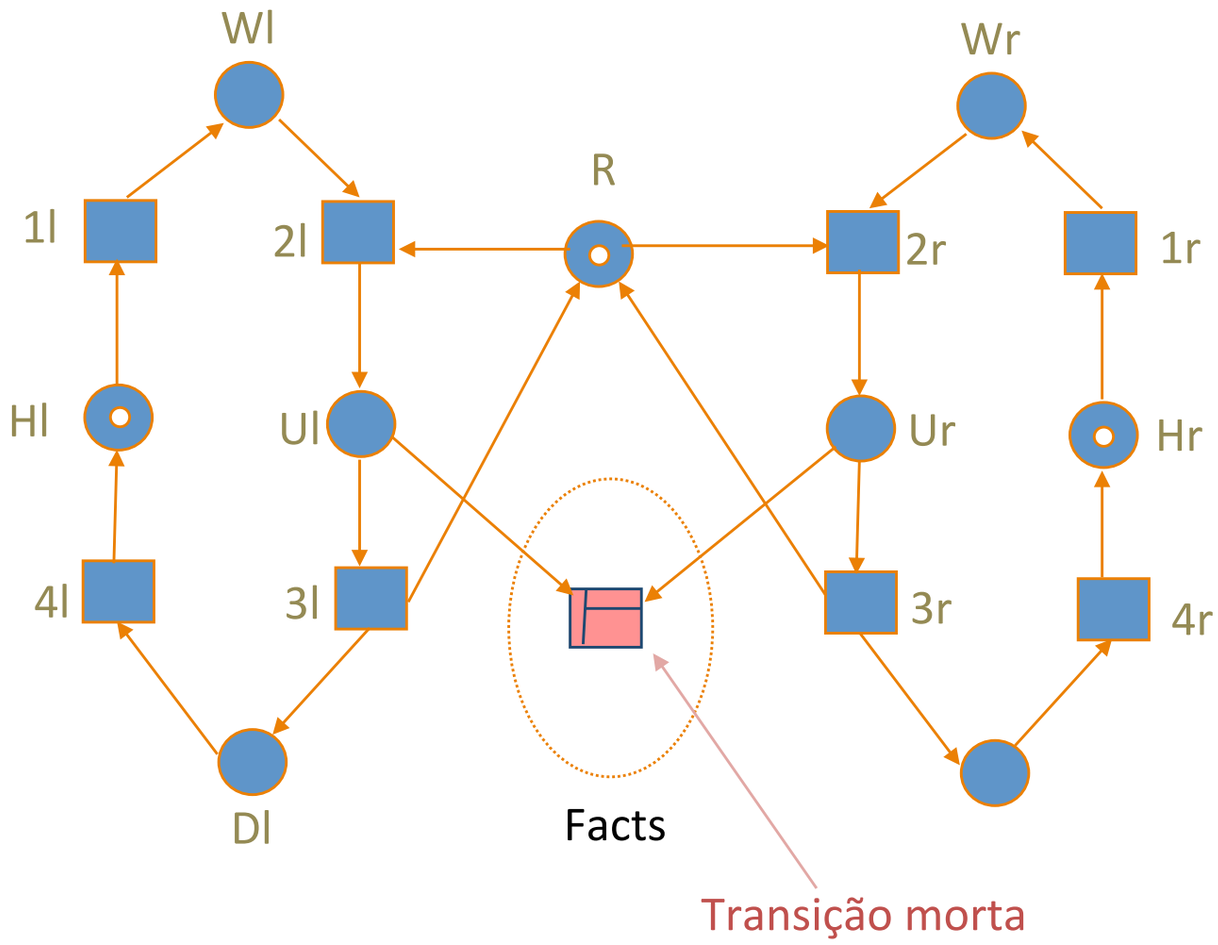
# Fatoração

A este processo de exploração das simetrias de uma rede para produzir uma nova rede dinamicamente equivalente à anterior mas de tamanho menor dá-se o nome de fatoração.

A rede resultante da fatoração é **chamada** de rede-quociente.

# Gerenciando processos de leitura e escrita em um computador

Em um sistema computacional os processos de leitura e escrita não podem ser feitos ao mesmo tempo de modo que se temos jobs que demandam leitura e escrita simultaneamente um semáforo deve gerenciar que ambos não sejam disparados ao mesmo tempo.

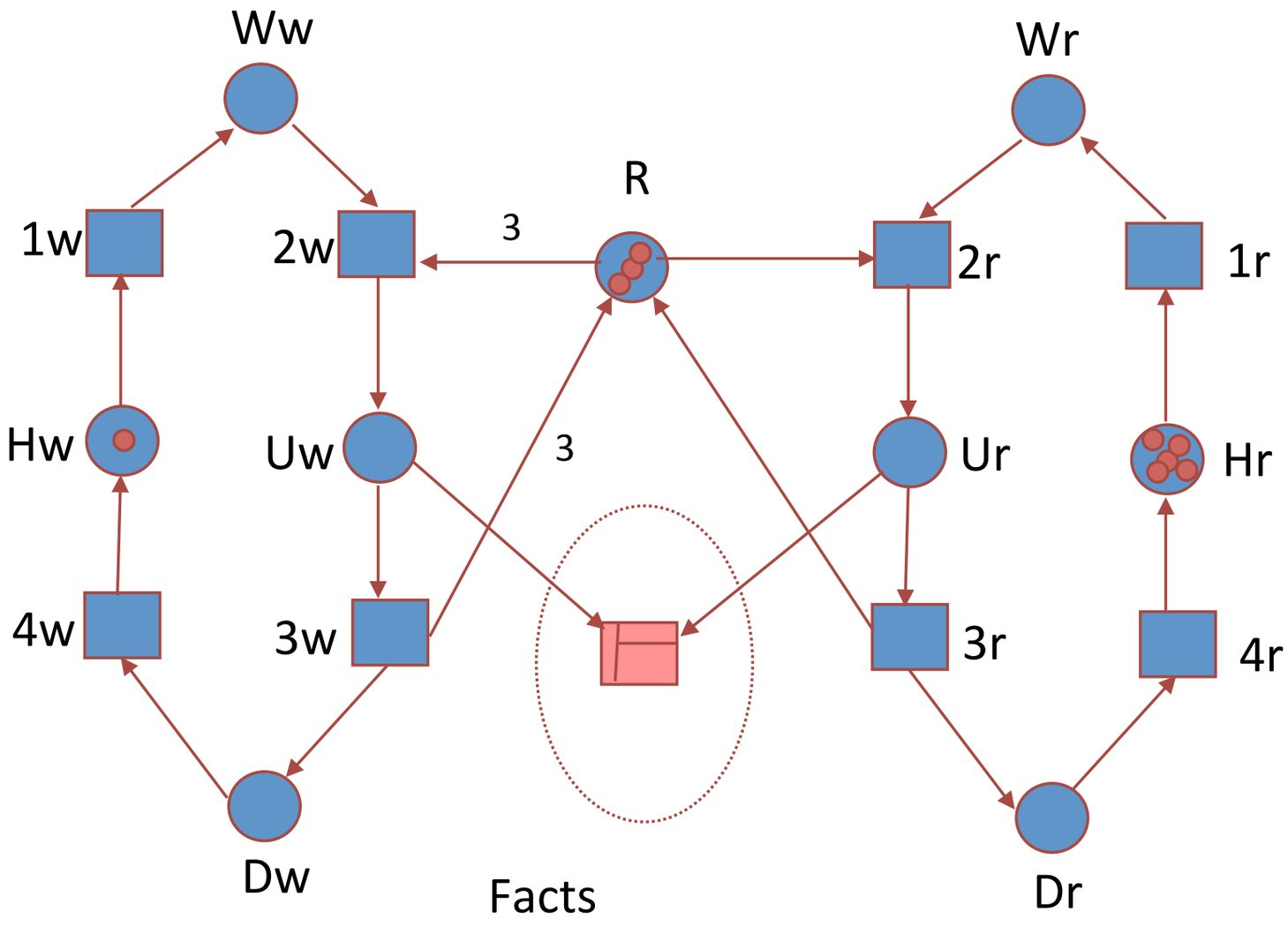


**Teorema 4**

Toda transição morta (deadlock) pode ser considerada um invariante e, reciprocamente, todo deadlock pode ser representado por um conjunto de trasições mortas.

Genrich, H.J., Lautenbach, K.; System Modelling with High-Level Petri Nets, Theoretical Computer Science, 13, 109-135, 1981.

# Voltando ao exemplo



# Cálculo dos invariantes

| | 1w | 2w | 3w | 4w | 1r | 2r | 3r | 4r | Mo | *i* |
|---|---|---|---|---|---|---|---|---|---|---|
| Hw | -1 | | | 1 | | | | | 1 | |
| Ww | 1 | -1 | | | | | | | | |
| Uw | | 1 | -1 | | | | | | | 3 |
| Dw | | | 1 | -1 | | | | | | |
| R | -3 | | 3 | | -1 | | 1 | | 3 | 1 |
| Hr | | | | | -1 | | | 1 | 5 | |
| Wr | | | | | 1 | -1 | | | | |
| Ur | | | | | | 1 | -1 | | | 1 |
| Dr | | | | | | | 1 | -1 | | |

Da equação de estado temos que $\mathbf{A}^T\sigma = \Delta\mathbf{M}$.

Assim, para o invariante de lugar, tem-se que $\mathbf{i}^T(\mathbf{A}^T\sigma) = \mathbf{i}^T\Delta\mathbf{M}$.

Portanto, devemos escolher o $\mathbf{i}$ de modo que $\mathbf{i}^T\Delta\mathbf{M} = 0$, e consequentemente $\mathbf{i}^T(\mathbf{A}^T\sigma) = (\mathbf{i}^T\mathbf{A}^T)\sigma = 0$, onde a solução trivial é $\mathbf{i}^T\mathbf{A}^T = 0$.

O invariante de lugar é ortogonal a todas as localidades das transições da rede.

| | 1w | 2w | 3w | 4w | 1r | 2r | 3r | 4r | Mo | *i* |
|---|---|---|---|---|---|---|---|---|---|---|
| Hw | -1 | | | 1 | | | | | 1 | |
| Ww | 1 | -1 | | | | | | | | |
| Uw | | 1 | -1 | | | | | | | 3 |
| Dw | | | 1 | -1 | | | | | | |
| R | -3 | 3 | | | -1 | 1 | | | 3 | 1 |
| Hr | | | | | -1 | | | 1 | 5 | |
| Wr | | | | | 1 | -1 | | | | |
| Ur | | | | | | 1 | -1 | | | 1 |
| Dr | | | | | | | 1 | -1 | | |

$x_1=x_2=x_4$

$x_2=x_3-3x_5$

$x_6=x_7=x_9$

$x_5=x_8-x_7$

Fazendo $x_1=x_6=x_7=0$

$x_3 = 3x_5$

$x_8 = x_5$

$$\mathbf{i}^T M = \mathbf{i}^T M_0$$

$$3M(Uw) + M(R) + M(Ur) =$$

$$3M_0(Uw) + M_0(R) + M_0(Ur) = M_0(R) = 3$$

# O problema da distinguibilidade

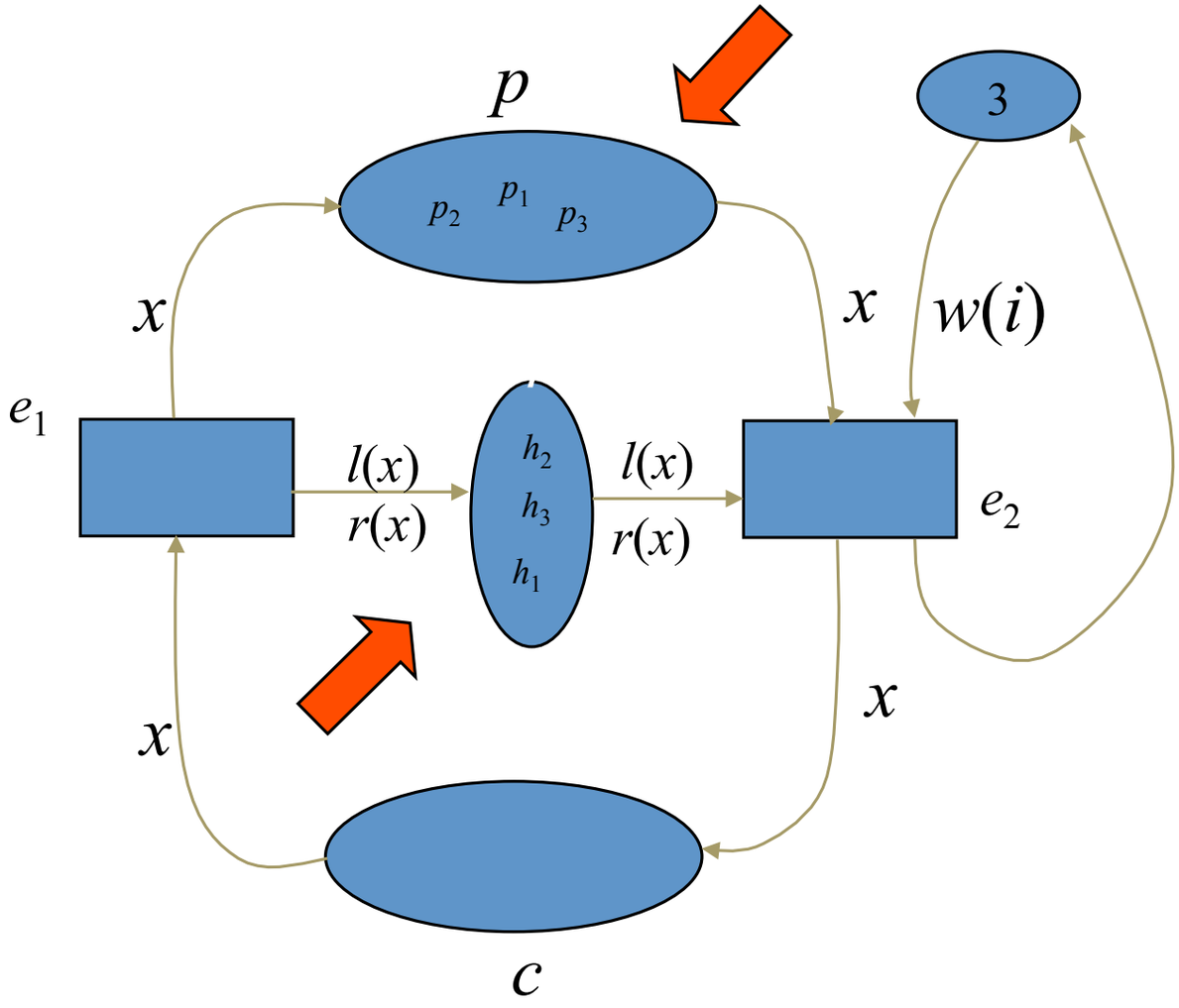
Como vimos anteriormente com o problema dos filósofos, a introdução da distinguibilidade das marcas (dobramento) não afeta as propriedades principais das redes.

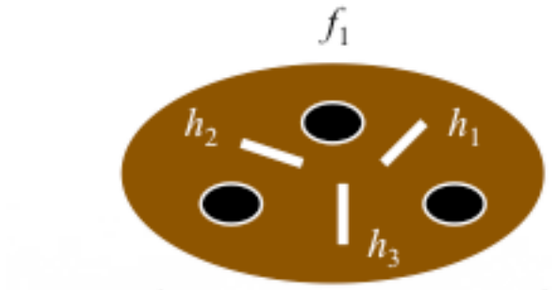
... isto é,

- preserva a dualidade
- preserva o conceito de localidade
- preserva o principio da concorrência
- preserva o principio da representação gráfica
- preserva o princípio da representação algébrica

Não saímos do domínio das redes de Petri

# Representação da marcação e multisets




A distinguibilidade das marcas nos leva à introdução de estruturas especiais no lugar das marcas, chamadas multisets.

# Multisets

Seja o conjunto (base set) {a,b,c,d,e,f,g,h}.

Em teoria de conjuntos, {a, c, f} ∪ { c, f, g}={a, c, f, g}.

Já em multisets (bags),

$$\{a, c, f\} \cup \{c, f, g\} = \{a, c, c, f, f, g\}$$

também descrito como 1`a + 2`c + 2`f + 1`g

De fato, o mesmo conjunto pode ser escrito como,

1`a + 0`b + 2`c + 0`d + 2`f + 1`g + 0`h

onde os coeficientes indicam o grau de multiplicidade de cada elemento do conjunto de base.

A forma simplificada para especificar este conjunto consiste em suprimir os elementos de coeficiente nulo.

## Operações básicas

***União*** : Sejam dois multisets representados pelos respectivos vetores de coeficientes sobre um conjunto de base (base set),

m= ($m_1$, $m_2$, ..., $m_i$, ..., $m_s$),
n= ($n_1$, $n_2$, ..., $n_j$, ..., $n_s$)

A união entre estes multisets é dada por,

$m \cup n = ( m_1 + n_1, \ldots m_i, n_j, \ldots, m_s + n_s)$

***Multiplicação por um escalar*** : Dado um multiset sobre um conjunto de base S, representado pelo seu vetor de coeficientes,

m= ($m_1$, ..., $m_i$, ..., $m_s$).

Está definida a multiplicação de m por um escalar *p*, resultando no seguinte multiset,

$p$ * m = ($p\ m_1$, ..., $p\ m_i$, ..., $p\ m_s$)

***Relação de Ordem parcial*** : Dados dois multisets, representados pelos respectivos vetores de coeficientes sobre um conjunto de base (base set) S,

m= ($m_1$, $m_2$, ..., $m_i$, ..., $m_s$),

n= ($n_1$, $n_2$, ..., $n_j$, ..., $n_s$)

Está definida a comparação entre estes dois multisets

m ≥ n se e somente se $m_i \geq n_i$ para todo $i$

***Cardinalidade** de um multiset* : Um multiset

m= ($m_1$, $m_2$, …, $m_i$, …, $m_s$),

tem cardinalidade |m|= $\Sigma\ m_i$

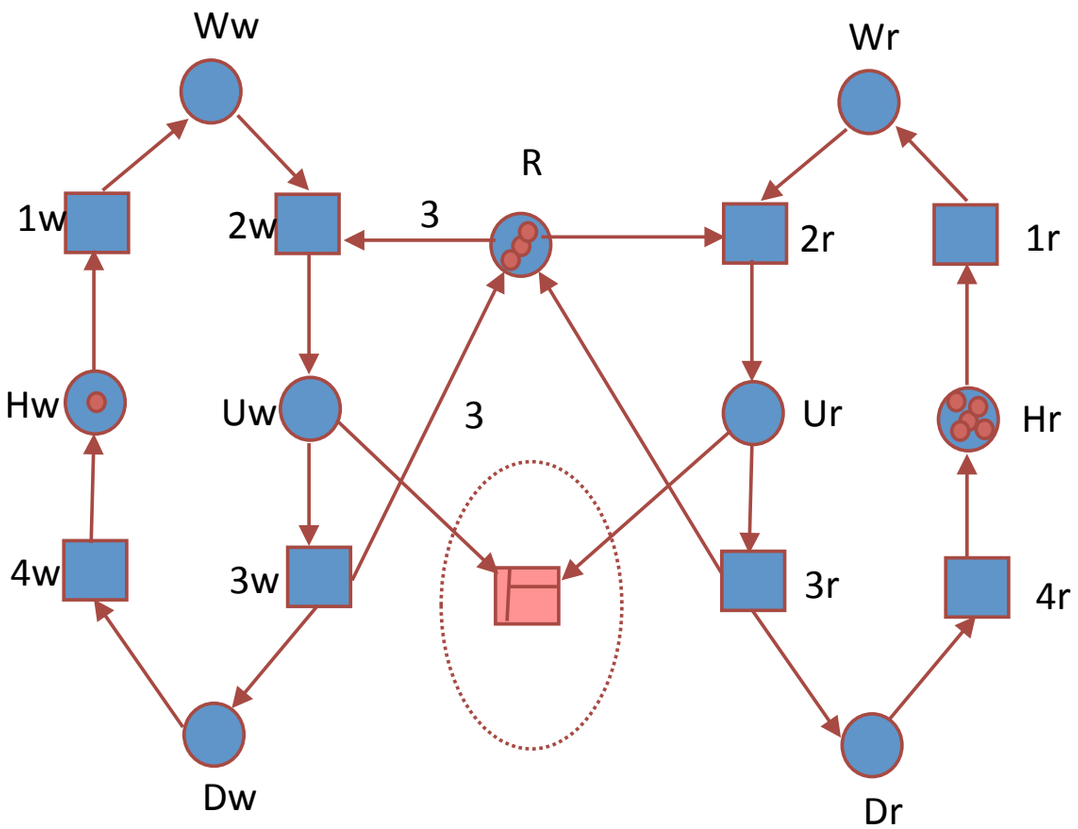
***Subtração*** : Sejam dois multisets representados pelos respectivos vetores de coeficientes sobre um conjunto de base (base set),

m= ($m_1$, $m_2$, ..., $m_i$, ..., $m_s$),
n= ($n_1$, $n_2$, ...,$n_i$, ..., $n_s$)

A subtração entre estes dois multisets, m - n, existe se m ≥ n e é dada por,

m - n = ($m_1$ - $n_1$, ..., $m_i$ - $n_i$, .... $m_s$ - $n_s$)

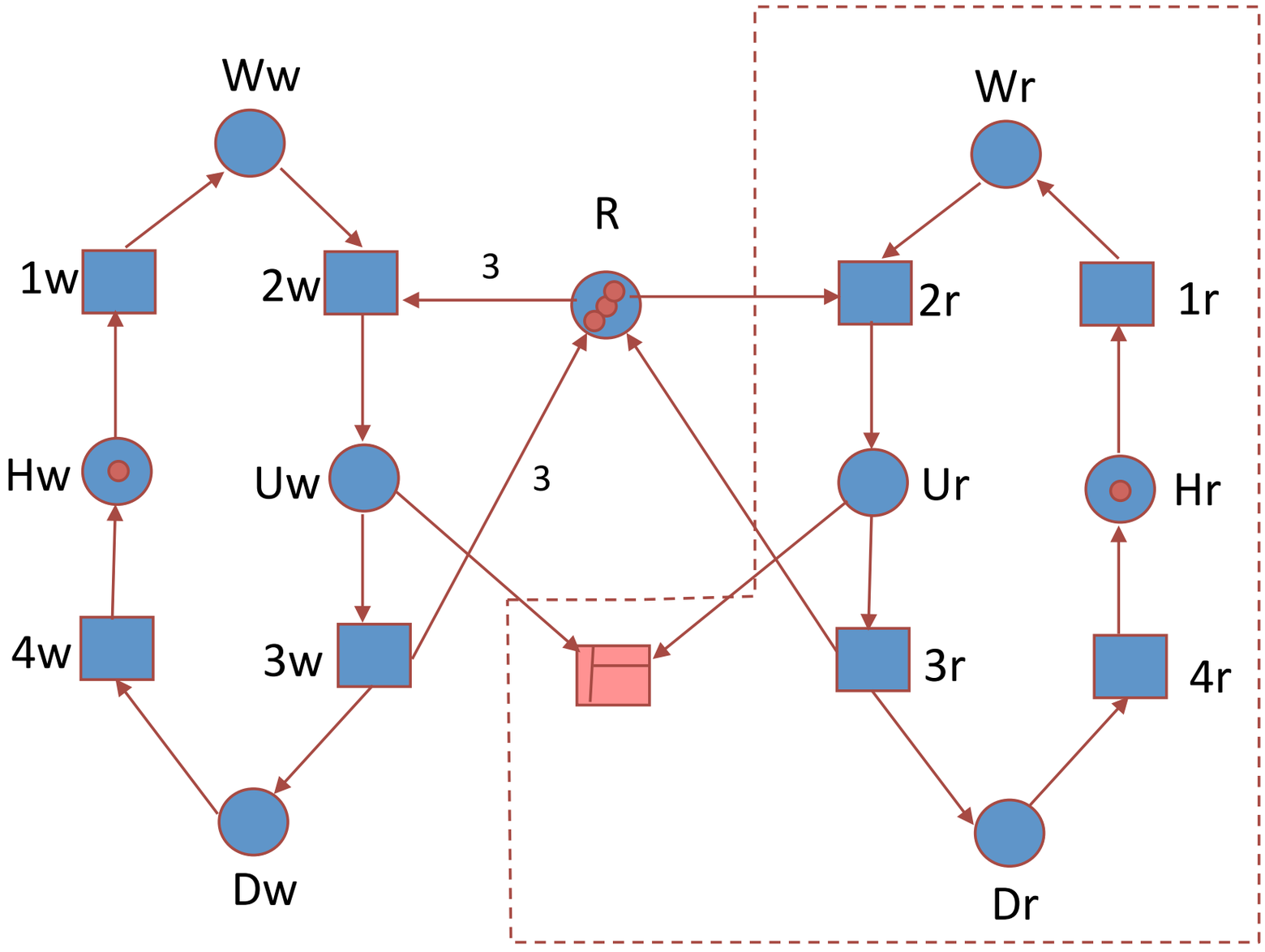
# Usando multisets




A rede do exemplo pode ser dobrada por simetria ou por replicação identificada.
Isto implica ou ter um único processo que pode representar leitura ou escrita ou
ter vários usuários de leitura (escrita) identificados.

Vamos então imaginar o problema de monitorar o serviço de leitura (escrita), onde os usuários pertencem ao conjunto C={a, b, c} e é possível funcionar em dois modos distintos: compartilhado (shared) ou exclusivo (exclusive).

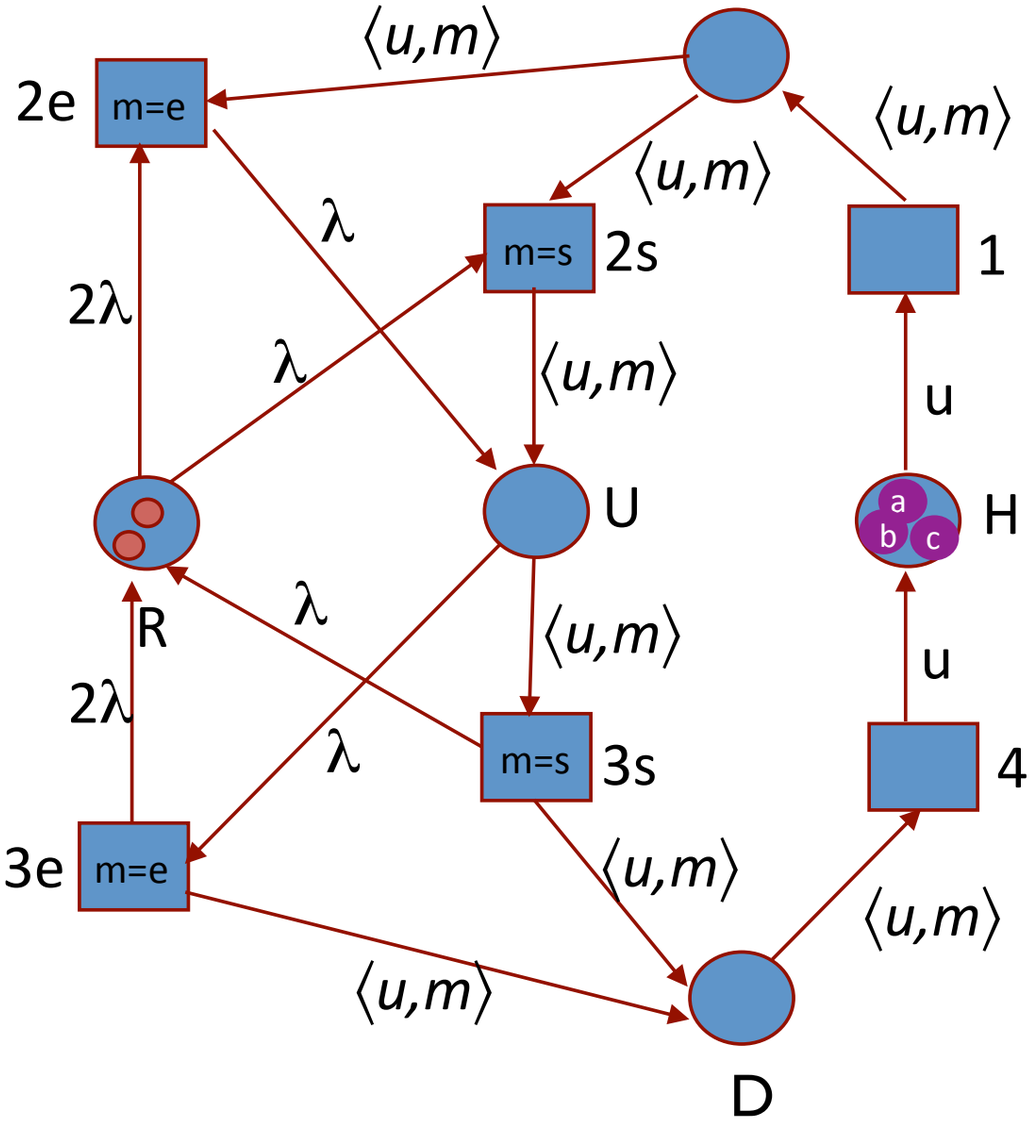
Assim, temos outro tipo M={ s, e}.


Ww
Wr
R
1w
2w
3
2r
1r
Hw
Uw
3
Ur
Hr
4w
3w
3r
4r
Dw
Dr

Introduzimos então os predicados,

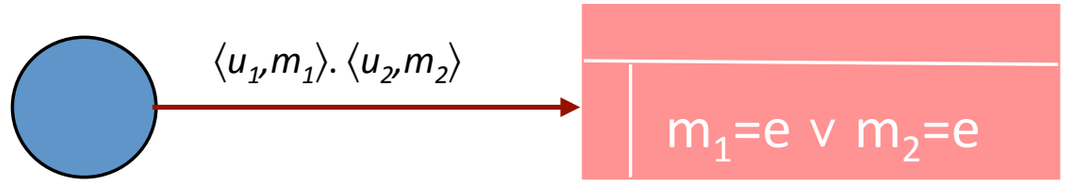
H⟨*u*⟩ : usuário *u* desocupado (hangling)
W⟨*u,m*⟩ : usuário *u* requisita recurso no modo *m*;
U⟨*u,m*⟩ : usuário *u* utilizando recurso no modo *m;*
D⟨*u,m*⟩ : usuário *u* terminou de utilizar o recurso no modo *m*.

R⟨ ⟩ : número de vêzes que o recurso está disponível



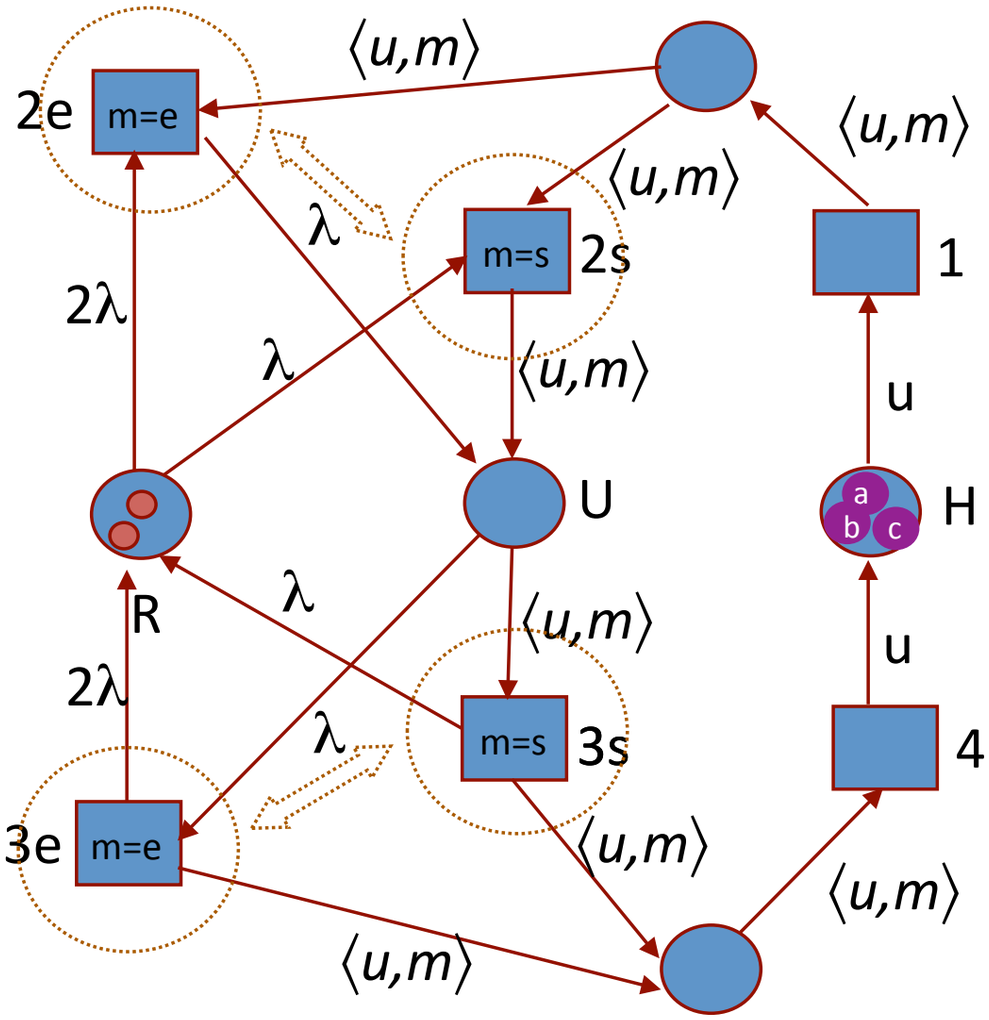
# Facts

O invariante mostrado anteriormente pode agora ser representado por outro predicado:

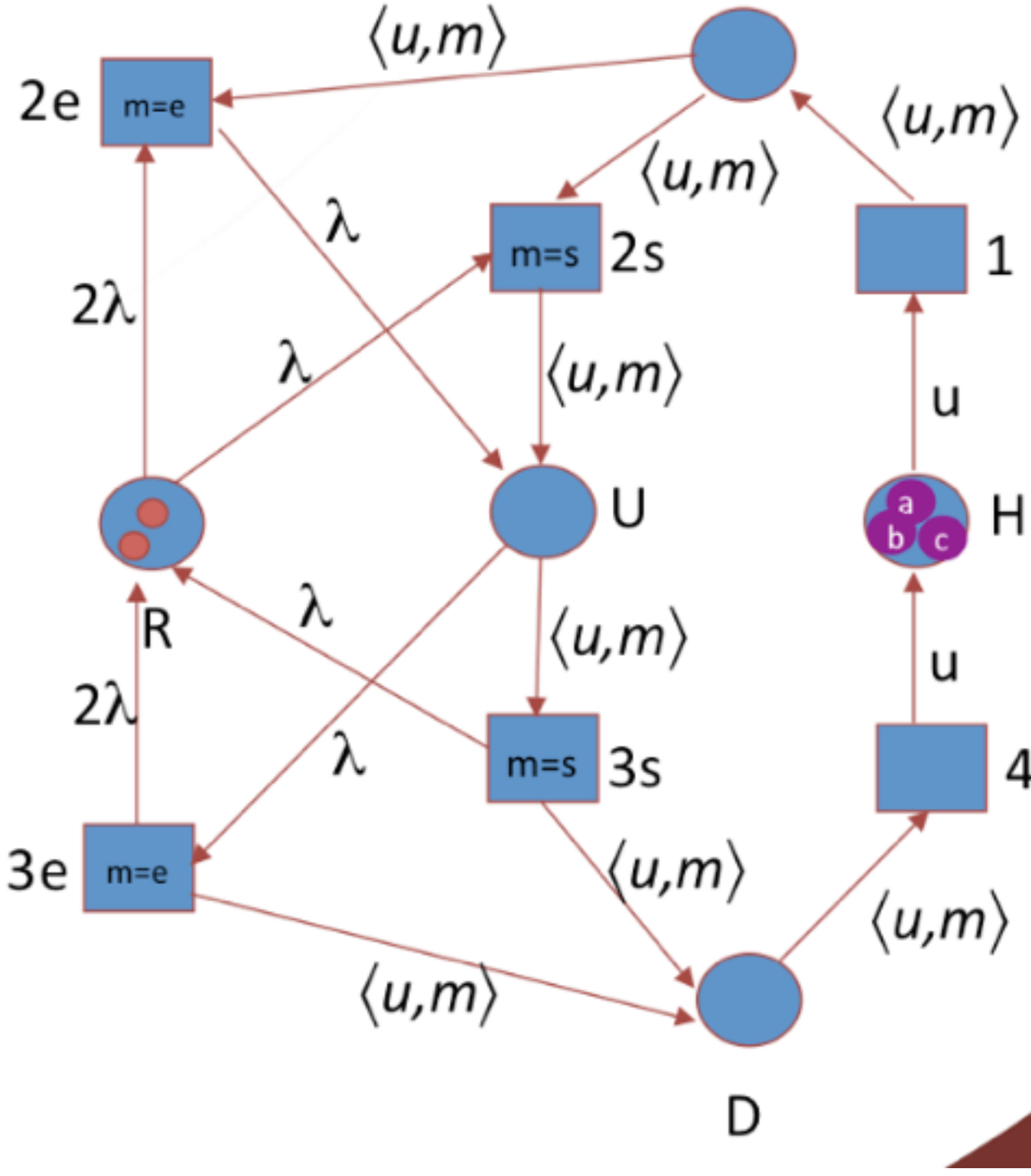



Teorema 4] Toda fórmula da lógica de primeira-ordem pode ser representada equivalentemente por transição morta de uma rede Pr/T com lugares de capacidade unitária (uma tupla não pode aparecer duas vezes em um dado lugar).

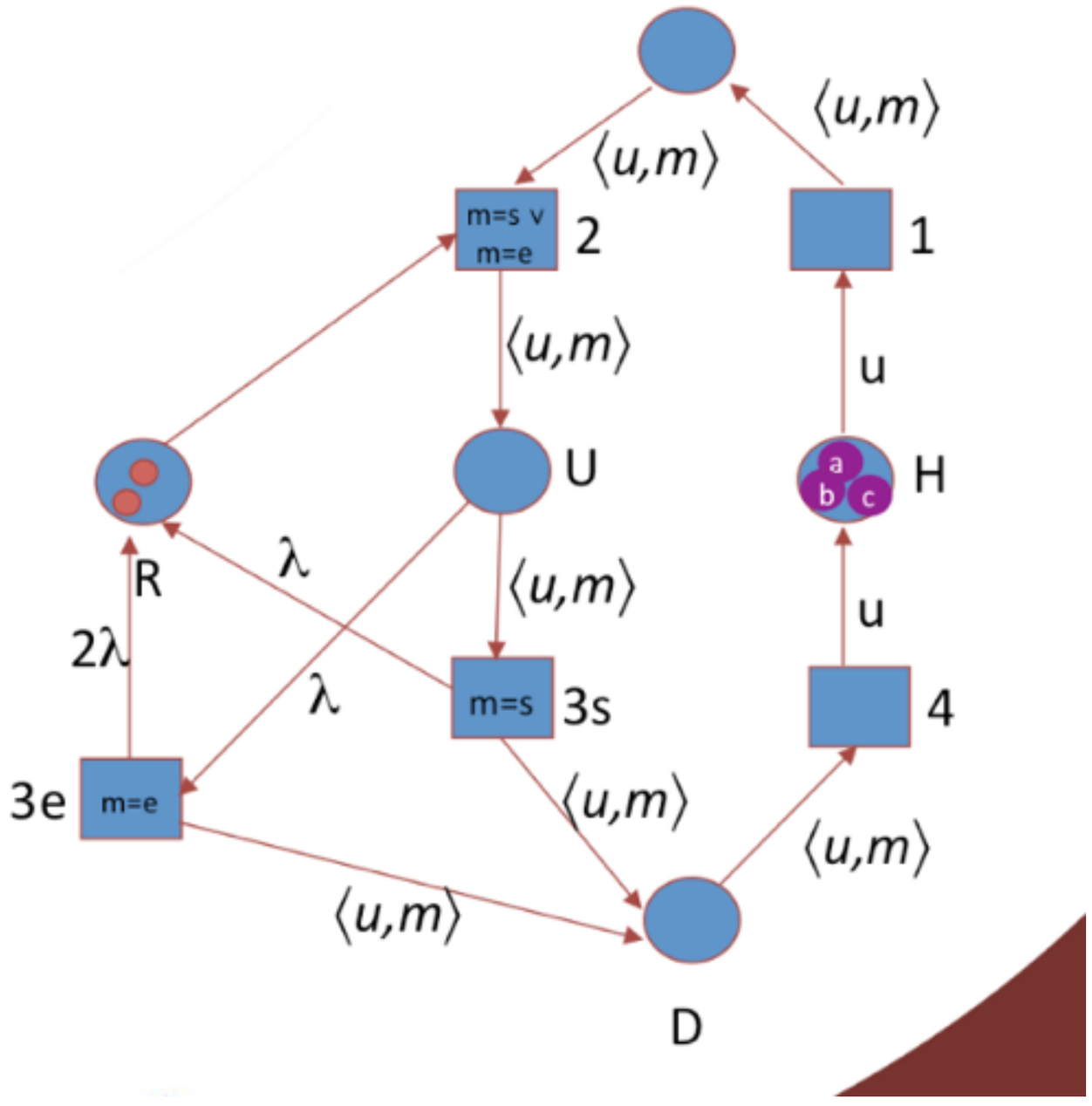
Seria possível reduzir ainda Mais a representação?



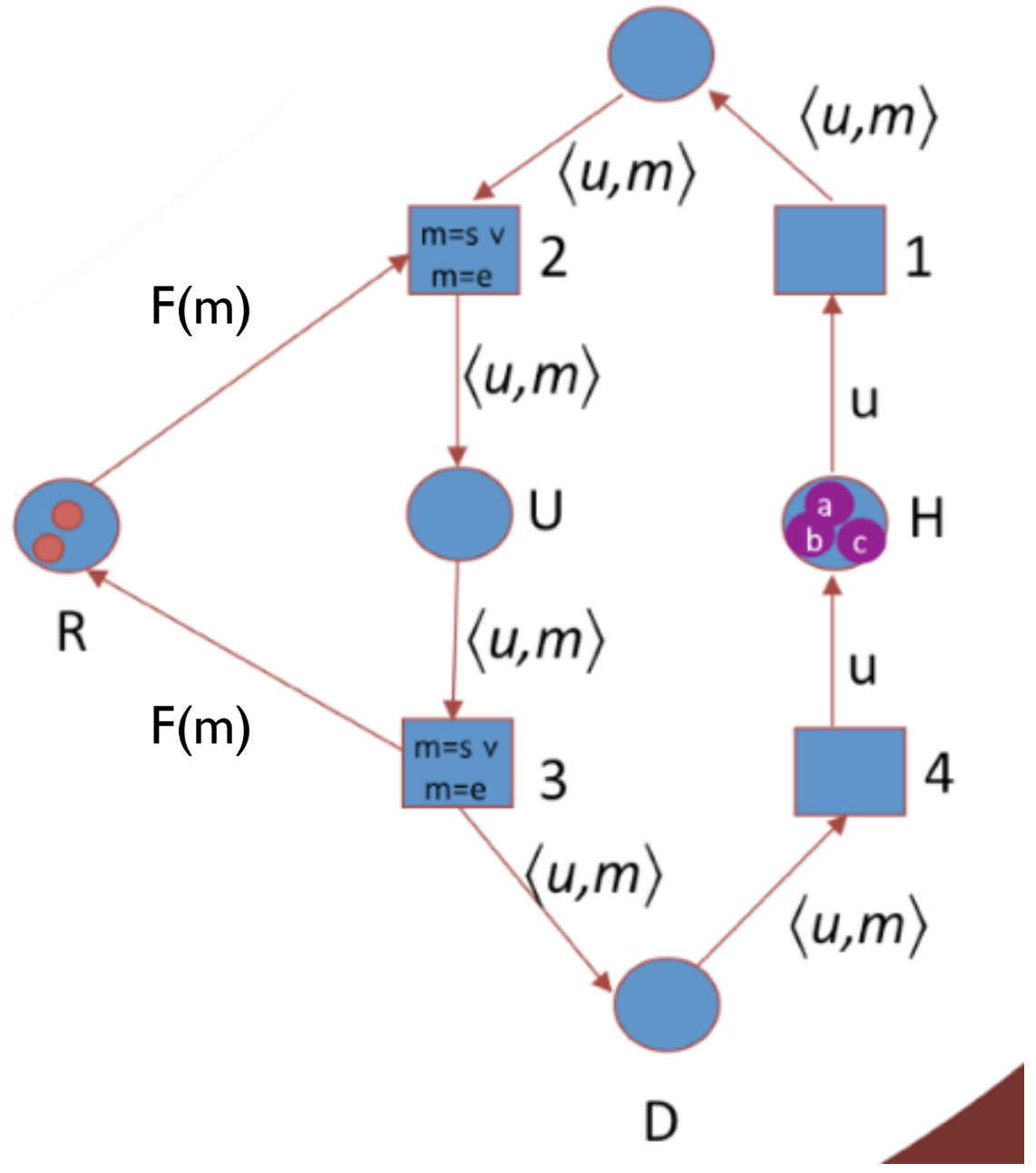
colapsando as transições 2e e 2s



colapsando agora as transições 3e e 3s...



$$F(m)=\begin{cases} 1\lambda, & m=s \\ 2\lambda, & m=e \end{cases}$$



## Definition 33

Uma rede Predicado-transição PrT, $(S, T; F, \Sigma, \mu, \Im, M_0, K)$ consiste de:
1) Uma estrutura de rede elementar $(S, T; F)$ onde as transições $T$ passam a representar predicados;
2) Uma estrutura $\Sigma$ composta por tipos (sorts), operadores e relações;
3) Um mapeamento $\mu$ de expressões bem formadas baseadas em multisets, dos elementos descritos acima no conjunto dos arcos da estrutura elementar, onde o comprimento da expressão está relacionado com a aridade dos predicados, e a aridade zero é representada pelo símbolo $\lambda$;
4) Inscrições $\Im$ compostas de fórmulas e predicados lógicos construídos com base nas operações e relações sobre a estrutura $\Sigma$, sendo que variáveis livres devem necessariamente aparecer no arco adjacente;
5) Uma marcação $M_0$ composta de predicados de $S$ definidos sobre os multisets dos elementos referidos em (2);
6) Um mapeamento $K$ sobre o conjunto $S$ que associa a cada predicado (lugar) um majorante $k$ do número de itens que este pode armazenar;

**Se a forma de reconhecer simetria for sempre “visual”, como alegam alguns autores, valeria a pena optar por uma representação em redes de alto nível?**

## Tipos de problemas

- Redes clássicas
- Redes de alto nível

## Não existe sistema fisicamente isolado

A relação do sistema com o seu ***contexto*** pode ser passiva ou ativa :

- passiva - o sistema troca apenas informação com o contexto
- ativa - o sistema interage com o contexto através de eventos internos, controláveis, e externos, que podem ser previstos mas não controlados por este.

# O dilema do método

Estruturação do problema e da representação
+
disciplina hierárquica (Van der Alst, Silva, Santos Fo.)

X

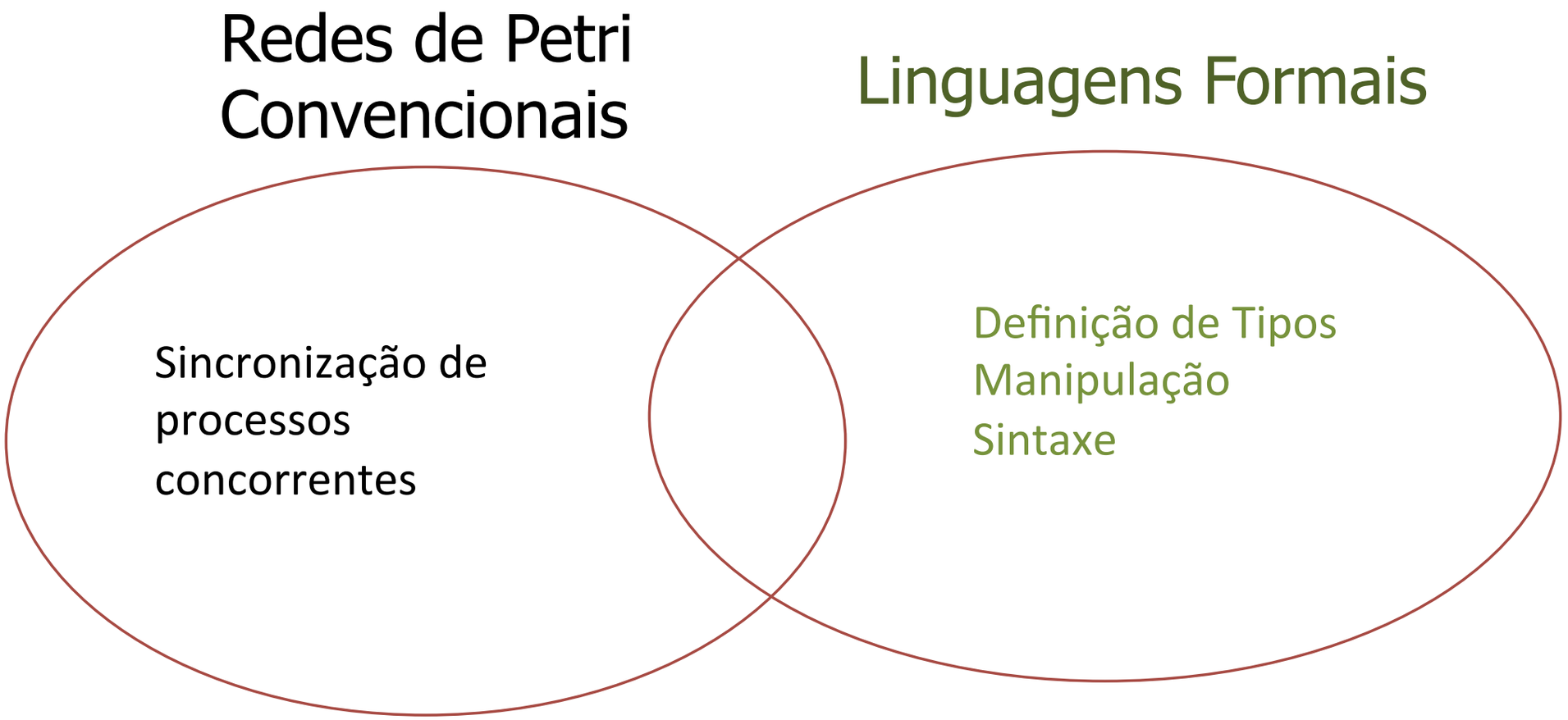
Organização por sub-sistemas autônomos cooperativos. Abordagem orientada a objetos. (Silberstin Blanc Lakos, Bastide,Gudwin, Silva).

# As alternativas

- Redes Coloridas  (baseadas em conjuntos e na linguagem ML)
- Redes orientadas a objetos
- Redes estendidas

# Redes Coloridas

- As marcas são divididas em conjuntos e separadas por tipo
- A área de declaração do sistema contém a identidade de cada variável assim como em declarações em ML
- Os arcos possuem inscrições e filtros que selecionam o tipo de marca que pode fluir por este arco.
- O comportamento dinâmico é dado pelo conjunto : grafo, inscrições e declaração.




**Kurt Jensen, An Introduction to the Practical Use of Coloured Petri Nets, Lect. Notes in Comp. Science 1492, 1998.**

Kurt Jensen
Univ. of Ahrus, Demark

# An Introduction to the Practical Use of Coloured Petri Nets

Kurt Jensen
Department of Computer Science, University of Aarhus
Ny Munkegade, Bldg. 540, DK-8000 Aarhus C, Denmark
Phone: +45 89 42 32 34, Telefax: +45 89 42 32 55
E-mail: kjensen@daimi.aau.dk, WWW: http://www.daimi.aau.dk/~kjensen/

**Abstract:** The development of Coloured Petri Nets (CP-nets or CPN) has been driven by the desire to develop a modelling language – at the same time theoretically well-founded and versatile enough to be used in practice for systems of the size and complexity found in typical industrial projects. To achieve this, we have combined the strength of Petri nets with the strength of programming languages. Petri nets provide the primitives for describing synchronisation of concurrent processes, while programming languages provide the primitives for definition of data types and manipulation of their data values.

The paper focuses on the practical use of Coloured Petri Nets. It introduces the basic ideas behind the CPN language, and it illustrates how CPN models can be analysed by means of simulation, state spaces and condensed state spaces. The paper

Fim