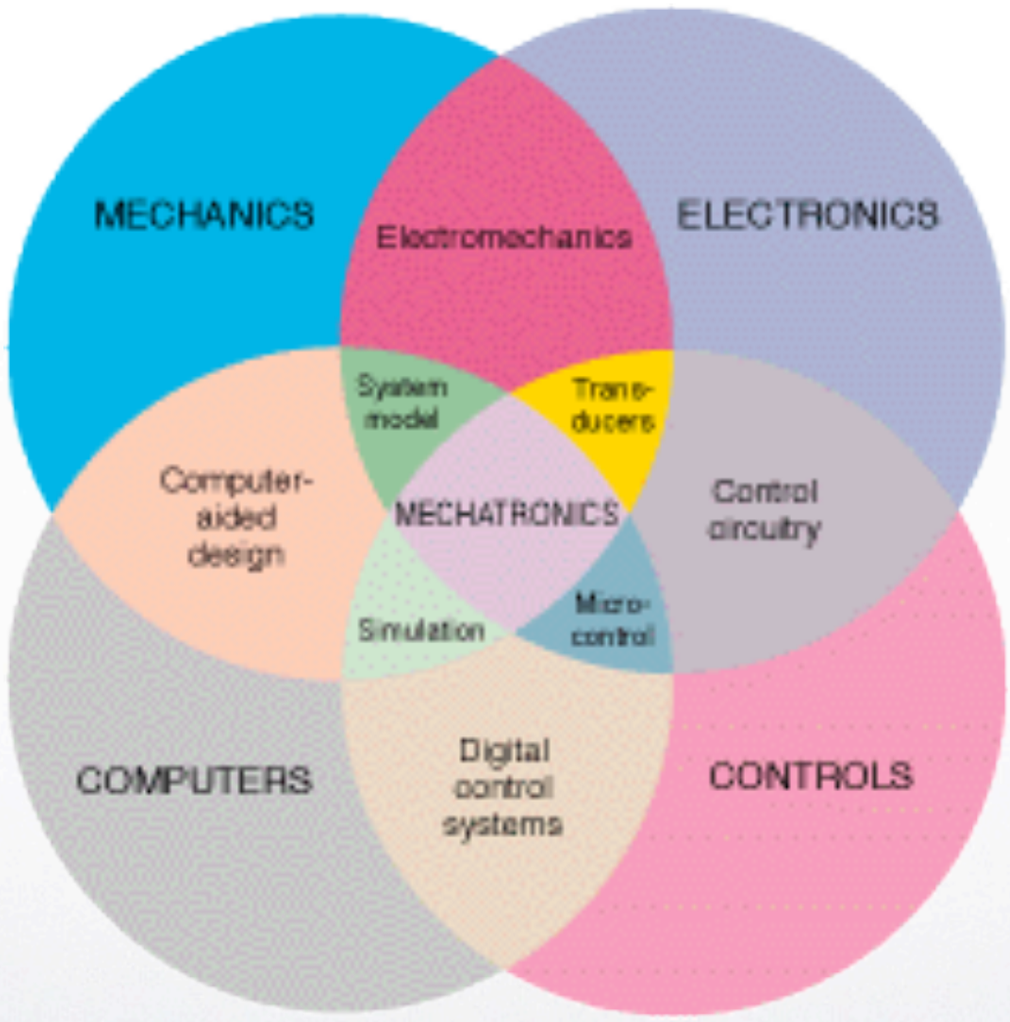
PMR 5237

Modelagem e Design de Sistemas Discretos em Redes de Petri

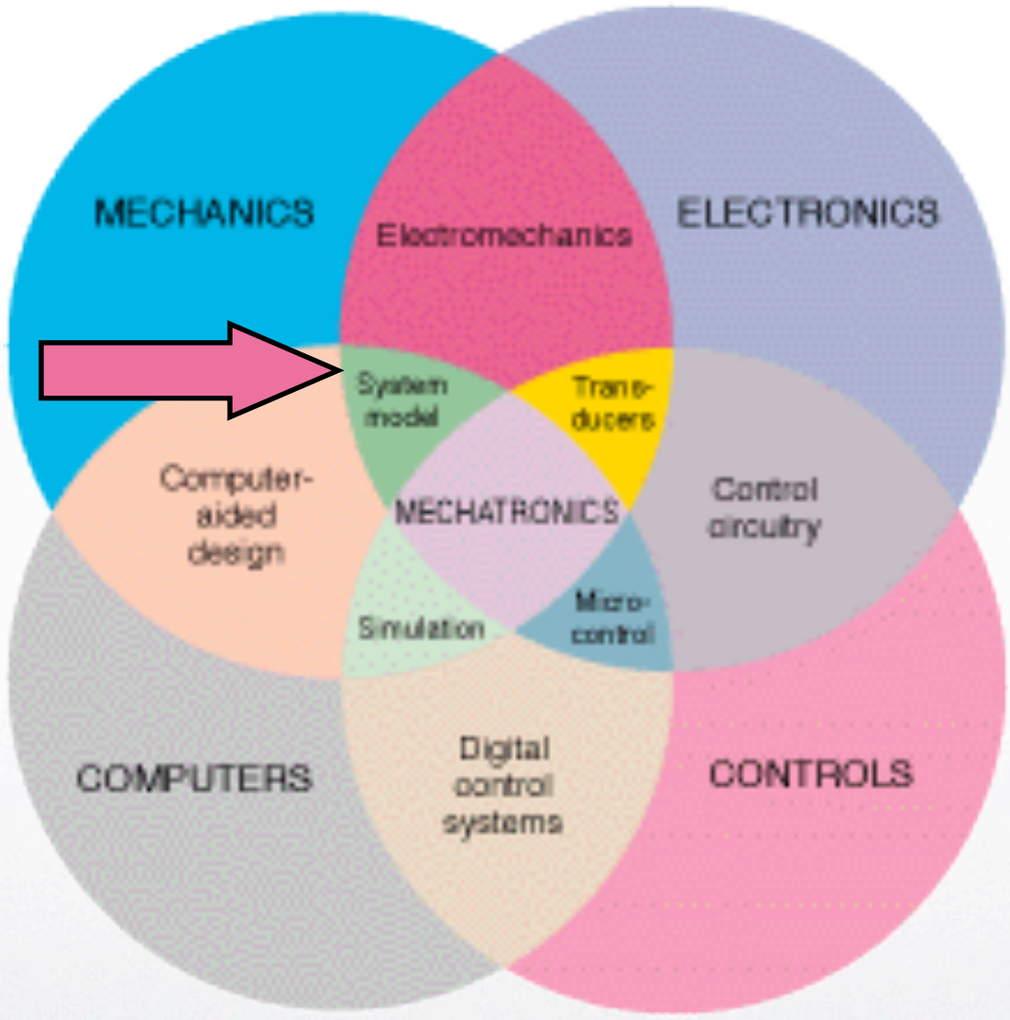
Aula 1: Introdução

Prof. José Reinaldo Silva
reinaldo@poli.usp.br

Contexto da Disciplina



Contexto da Disciplina



Histórico das RdP

Criada em 1962 com a tese de doutorado de Carl Adam Petri (*Communication with Automata*)

modelagem de processos

Carl Adam Petri nasceu em Leipzig, em 12 de julho de 1926 e morreu aos 83 anos em 2 de Julho de 2010, deixando um dos maiores legados teórico-científicos do século XX, desenvolvido em pouco mais de 40 anos.

2001 UML 2.0 modelagem de requisitos

modelagem de workflow

Histórico das RdP

Carl Adam Petri nasceu em Leipzig, em 12 de julho de 1926 e morreu aos 83 anos em 2 de Julho de 2010, deixando um dos maiores legados teórico-científicos do século XX, desenvolvido em pouco mais de 40 anos.

Criada em 1962 com a tese de doutorado de Carl Adam Petri (*Communication with Automata*)

modelagem de processos

2001 UML 2.0 modelagem de requisitos

modelagem de workflow

Histórico das RdP

Carl Adam Petri nasceu em Leipzig, em 12 de julho de 1926 e morreu aos 83 anos em 2 de Julho de 2010, deixando um dos maiores legados teórico-científicos do século XX, desenvolvido em pouco mais de 40 anos.

Criada em 1962 com a tese de doutorado de Carl Adam Petri (*Communication with Automata*)

modelagem de processos

2001 UML 2.0 modelagem de requisitos

modelagem de workflow

Histórico das RdP

Criada em 1962 com a tese de doutorado de Carl Adam Petri (*Communication with Automata*)

Carl Adam Petri nasceu em Leipzig, em 12 de julho de 1926 e morreu aos 83 anos em 2 de Julho de 2010, deixando um dos maiores legados teórico-científicos do século XX, desenvolvido em pouco mais de 40 anos.

modelagem de processos

2001 UML 2.0 modelagem de requisitos

modelagem de workflow

Histórico das RdP

Carl Adam Petri nasceu em Leipzig, em 12 de julho de 1926 e morreu aos 83 anos em 2 de Julho de 2010, deixando um dos maiores legados teórico-científicos do século XX, desenvolvido em pouco mais de 40 anos.

Criada em 1962 com a tese de doutorado de Carl Adam Petri (*Communication with Automata*)

modelagem de processos

2001 UML 2.0 modelagem de requisitos

modelagem de workflow

Schemas

Histórico das RdP

Carl Adam Petri nasceu em Leipzig, em 12 de julho de 1926 e morreu aos 83 anos em 2 de Julho de 2010, deixando um dos maiores legados teórico-científicos do século XX, desenvolvido em pouco mais de 40 anos.

Criada em 1962 com a tese de doutorado de Carl Adam Petri (*Communication with Automata*)

modelagem de processos

2001 UML 2.0 modelagem de requisitos

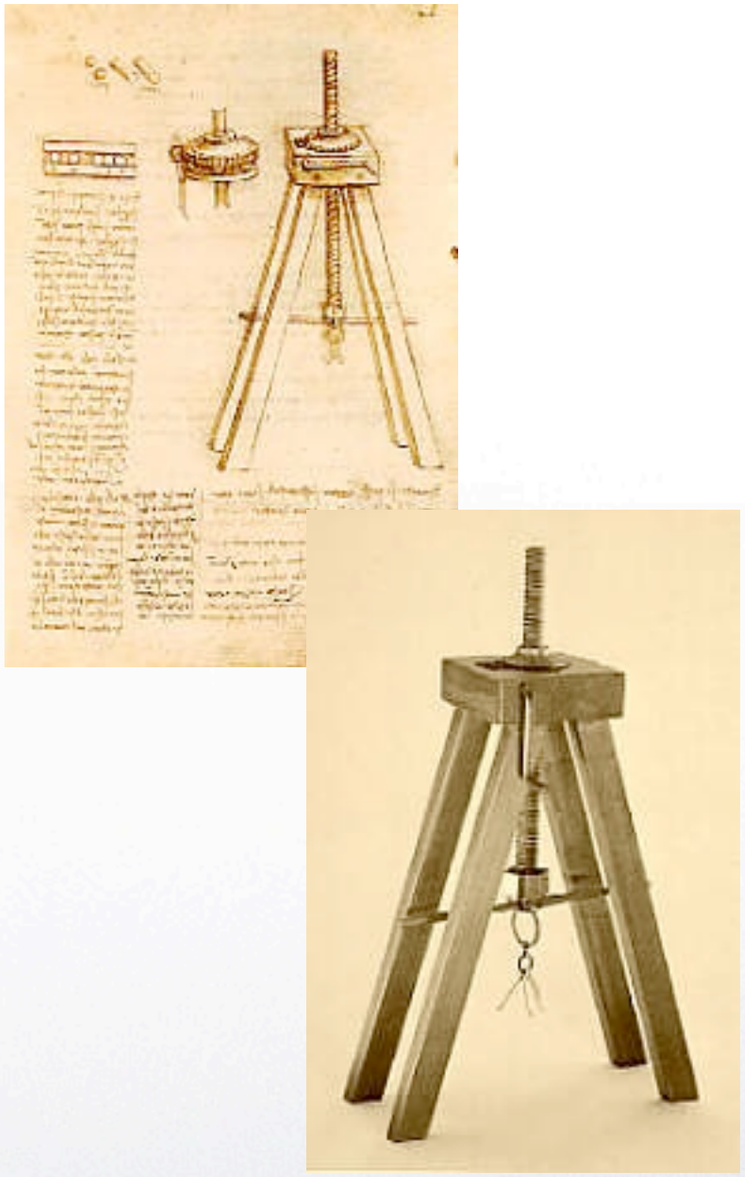
modelagem de workflow

Schemas

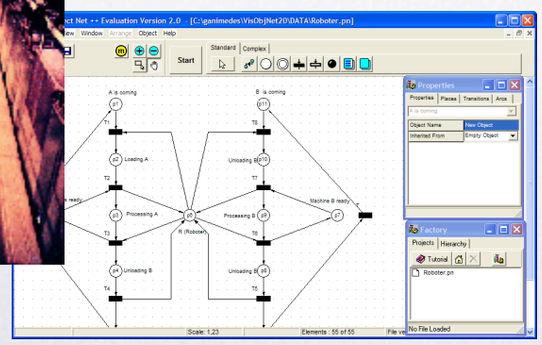
A schema is a cognitive framework or concept that helps organize and interpret information. Schemas can be useful because they allow us to take shortcuts in interpreting a vast amount of information

A “arte” de modelar

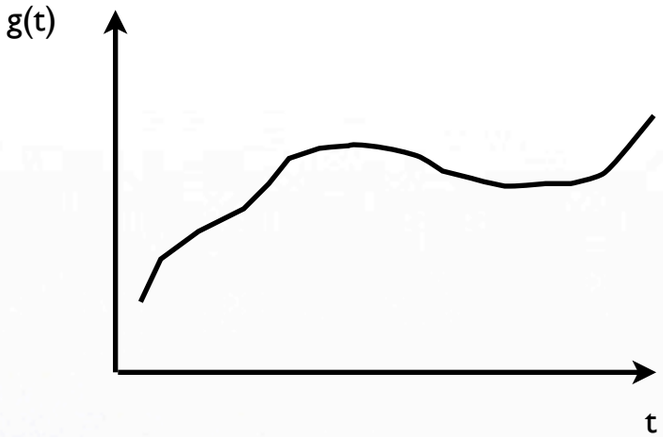
Modelar é produzir uma representação esquemática de um objeto ou sistema capturando suas propriedades mais importantes e a relação entre as partes dinâmmicas, geralmente de modo a compactar as informações, aumentar o nivel de abstração. Em Engenharia, a maior parte das vezes um modelo é o ponto de partida para a análise de um artefato que pode ser feita analiticamente ou através de simulação.

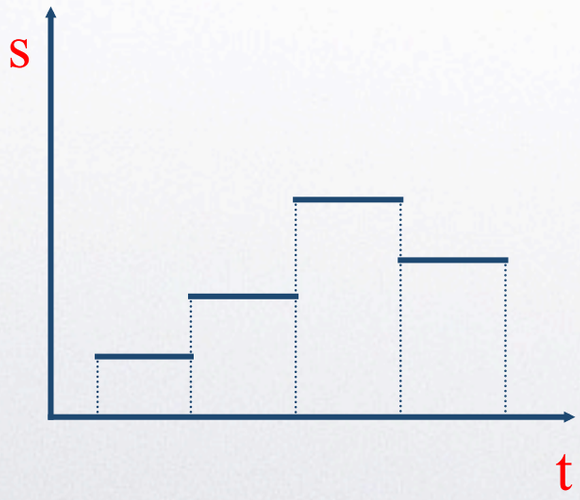


Modelo de Leonardo da Vinci



Paradigmas para modelagem: Estado/Transição





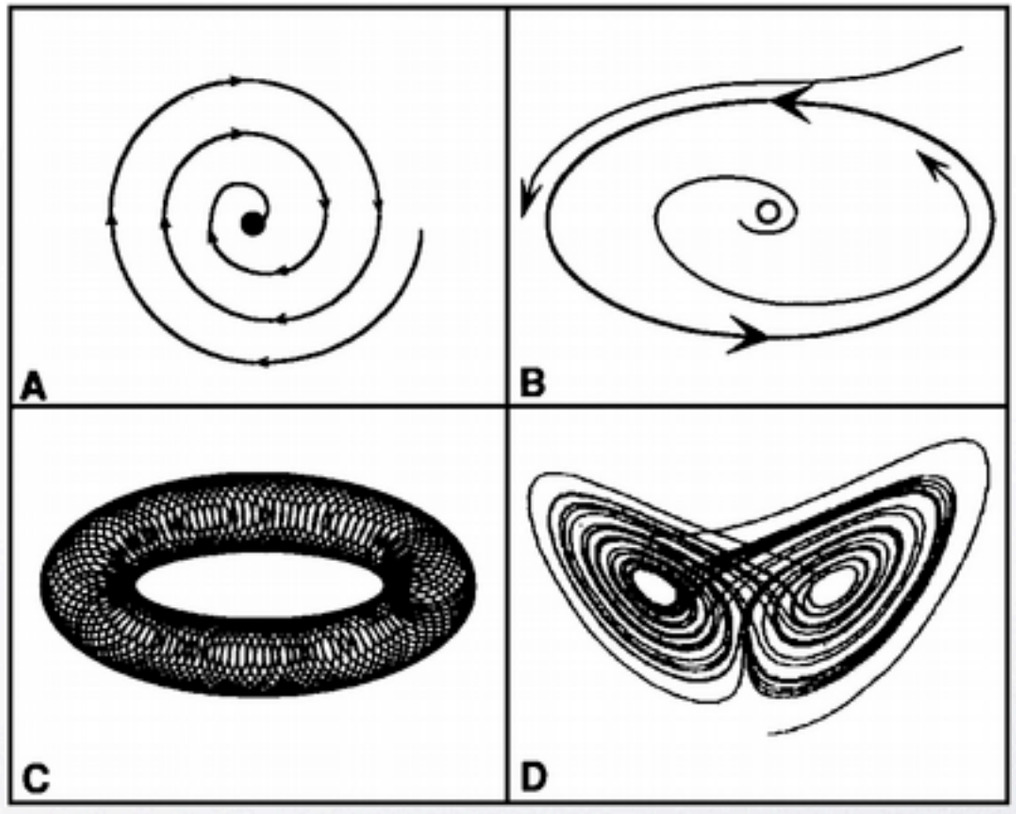
A descrição do comportamento de sitemas é feita identificando uma função das chamadas "variáveis de estado", ou variáveis que caracterizam o sistema. Este tem pontos de equilibrio que caracterizam o estado e evoluem no tempo denotando a característica do sistema. A evolução se dá quando "eventos" ocorrem.

Em algums casos a função de estado é contínua nas variáveis de estado e também no tempo caracterizando trajetórias também contínuas. Em outros casos a função de estado é discreta e evolui aos saltos caracterizando um sistema discreto.

Diagrama de Eventos

Vamos agora fazer uma inversão e imaginar que temos um sistema dinâmico *f(x)* e em um diagrama 2D onde cada ponto representa um evento. Aplicando-se cada um destes eventos a *f(x)* temos que o sistema muda de estado e está preparado para receber outro evento. A evolução do sistema evento a evento (não necessariamente ligada ao tempo, mas somente à sequência admissível de eventos) também dá uma trajetória no espaço de eventos.

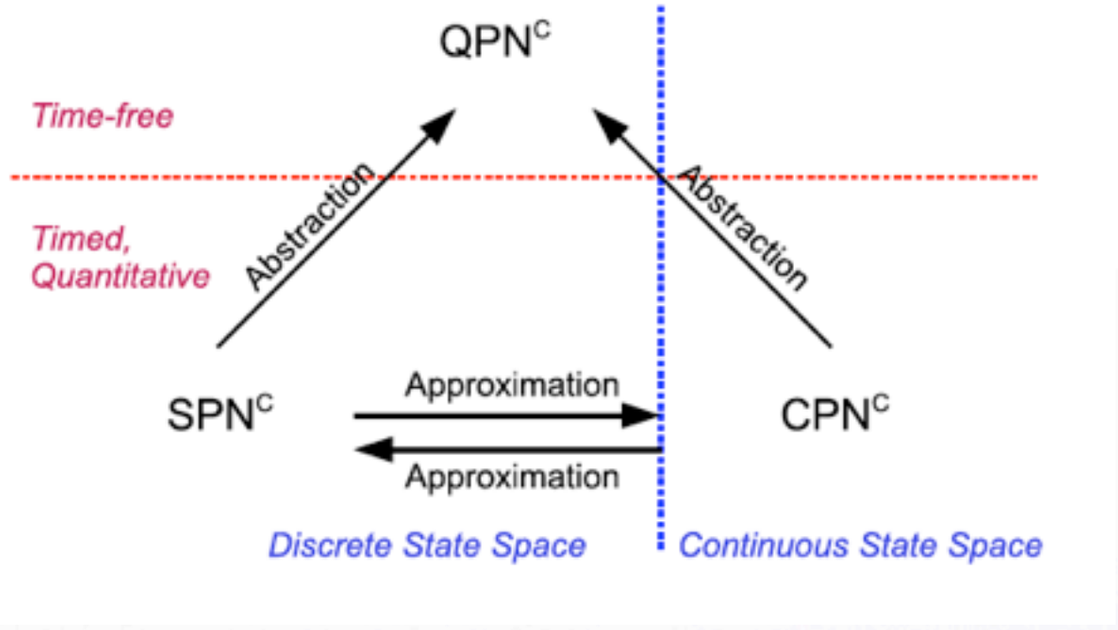
Eventualmente esta trajetória pode convergir para valores (sem necessariamente atingí-los) que são chamados atratores.



Novas Aplicações das Redes de Petri

Hoje as redes de Petri são aplicadas em um universo ainda mais amplo e muito mais abrangente do que a comunicação entre processos, redes de computadores, etc.

As novas aplicações mais abrangentes são feitas em sistemas complexos que podem ser associados, por exemplo a sistemas complexos de manufatura, "*harzardous systms*", sistemas de automação mais complexos e sistemas biologicos.

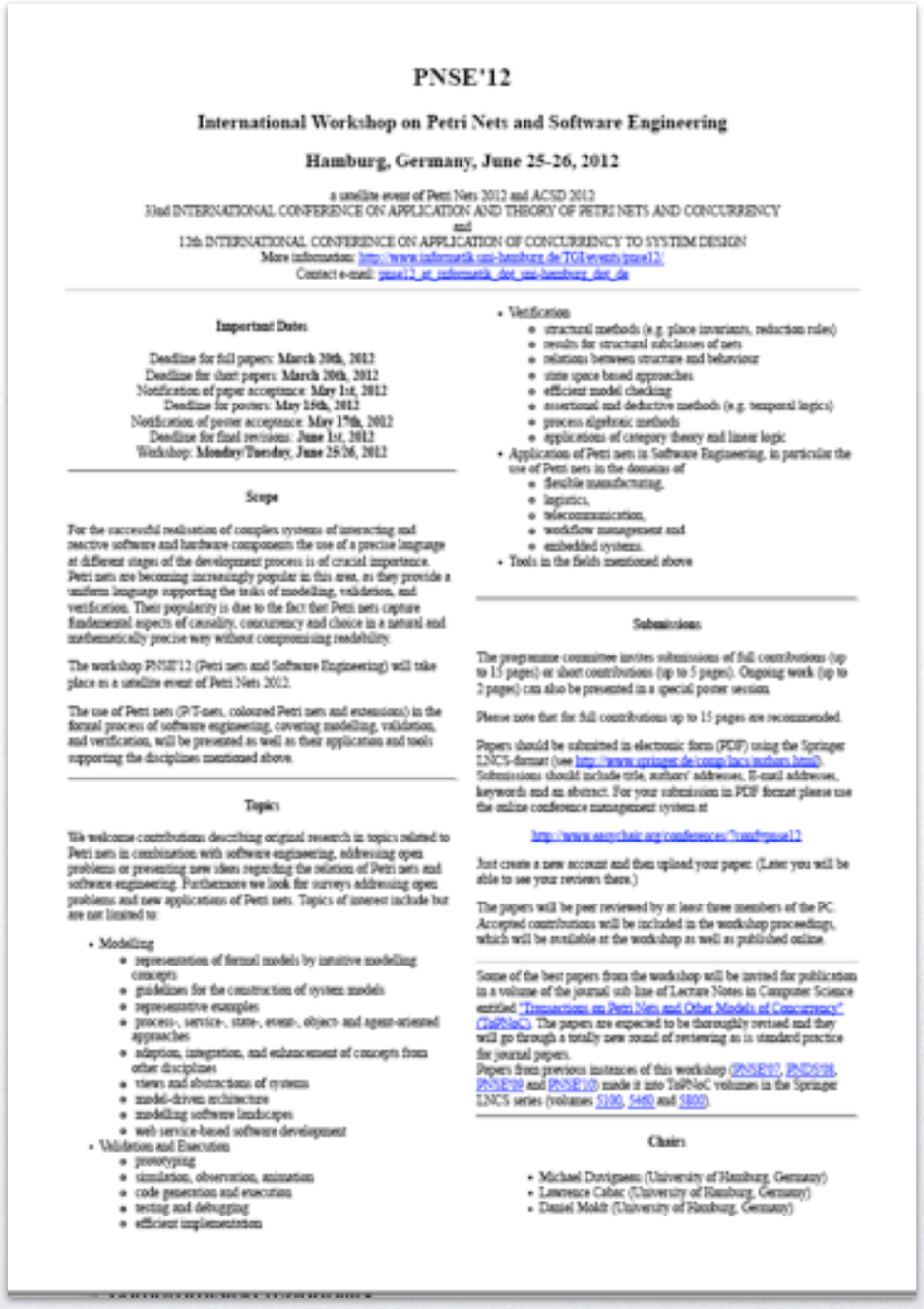


Liu, F. and Heiner, M.; Colored Petri Nets to Model and Simulate Biological Systems, Recent Advances in Petri Nets and Concurrency, S. Donatelli, J. Kleijn, R.J. Machado, J.M. Fernandes (eds.), CEUR Workshop Proceedings, volume 827, ISSN 1613-0073, Jan/2012, pp. 71-85.

Aplicações históricas e Tradicionais das RdP

Além da aplicação clássica em sistemas dinâmicos as Redes de Petri foram historicamente aplicadas em Engenharia de Software desde os anos 80 com Gerald Estrin na UCLA e Stephen Yau em Ilinois Urbana-Champagne. Hoje o congresso mais importante de Redes de Petri no mundo, o 33th Int. Conference on Application and Theory of Petri Nets and Cuncurrecy (realizada em conjunto com a 12th Int. Conference on Application of Concurrency to Systems Design), tem um workshop (desde 2008) dedicado especificamente a Petri Nets on Software Engineering, o PNSE. Ao lado está o Call for Papers desde workshop em 2012.

Outra aplicação que se tornou clássica é em Redes de Computadores, especialmente o roteamento inteligente praticado no projeto Superhighway.



PNSE'12

International Workshop on Petri Nets and Software Engineering

Hamburg, Germany, June 25-26, 2012

a satellite event of Petri Nets 2012 and ACSD 2012
33rd INTERNATIONAL CONFERENCE ON APPLICATION AND THEORY OF PETRI NETS AND CONCURRENCY
and
12th INTERNATIONAL CONFERENCE ON APPLICATION OF CONCURRENCY TO SYSTEM DESIGN
More information: http://www.informatik.uni-hamburg.de/TGI/events/pnse12/
Contact e-mail: pnse12_at_informatik_dot_uni-hamburg_dot_de

Important Dates

Deadline for full papers: **March 20th, 2012**
Deadline for short papers: **March 20th, 2012**
Notification of paper acceptance: **May 1st, 2012**
Deadline for posters: **May 15th, 2012**
Notification of poster acceptance: **May 17th, 2012**
Deadline for final revisions: **June 1st, 2012**
Workshop: **Monday/Tuesday, June 25/26, 2012**

Scope

For the successful realization of complex systems of interacting and reactive software and hardware components the use of a precise language at different stages of the development process is of crucial importance. Petri nets are becoming increasingly popular in this area, as they provide a uniform language supporting the tasks of modelling, validation, and verification. Their popularity is due to the fact that Petri nets capture fundamental aspects of causality, concurrency and choice in a natural and mathematically precise way without compromising readability.

The workshop PNSE'12 (Petri nets and Software Engineering) will take place as a satellite event of Petri Nets 2012.

The use of Petri nets (P/T-nets, coloured Petri nets and extensions) in the formal process of software engineering, covering modelling, validation, and verification, will be presented as well as their application and tools supporting the disciplines mentioned above.

Topics

We welcome contributions describing original research in topics related to Petri nets in combination with software engineering, addressing open problems or presenting new ideas regarding the relation of Petri nets and software engineering. Furthermore we look for surveys addressing open problems and new applications of Petri nets. Topics of interest include but are not limited to:

- Modelling
  - representation of formal models by intuitive modelling concepts
  - guidelines for the construction of system models
  - representative examples
  - process-, service-, state-, event-, object- and agent-oriented approaches
  - adaption, integration, and enhancement of concepts from other disciplines
  - views and abstractions of systems
  - model-driven architecture
  - modelling software landscapes
  - web service-based software development
- Validation and Execution
  - prototyping
  - simulation, observation, animation
  - code generation and execution
  - testing and debugging
  - efficient implementation
- Verification
  - structural methods (e.g. place invariants, reduction rules)
  - results for structural subclasses of nets
  - relations between structure and behaviour
  - state space based approaches
  - efficient model checking
  - assertional and deductive methods (e.g. temporal logics)
  - process algebraic methods
  - applications of category theory and linear logic
- Application of Petri nets in Software Engineering, in particular the use of Petri nets in the domains of
  - flexible manufacturing,
  - logistics,
  - telecommunication,
  - workflow management and
  - embedded systems.
- Tools in the fields mentioned above

Submissions

The programme committee invites submissions of full contributions (up to 15 pages) or short contributions (up to 5 pages). Ongoing work (up to 2 pages) can also be presented in a special poster session.

Please note that for full contributions up to 15 pages are recommended.

Papers should be submitted in electronic form (PDF) using the Springer LNCS-format (see http://www.springer.de/comp/lncs/authors.html). Submissions should include title, authors' addresses, E-mail addresses, keywords and an abstract. For your submissions in PDF format please use the online conference management system at

http://www.easychair.org/conferences/?conf=pnse12

Just create a new account and then upload your paper. (Later you will be able to see your reviews there.)

The papers will be peer reviewed by at least three members of the PC. Accepted contributions will be included in the workshop proceedings, which will be available at the workshop as well as published online.

Some of the best papers from the workshop will be invited for publication in a volume of the journal sub line of Lecture Notes in Computer Science entitled "Transactions on Petri Nets and Other Models of Concurrency" (ToPNoC). The papers are expected to be thoroughly revised and they will go through a totally new round of reviewing as is standard practice for journal papers.
Papers from previous instances of this workshop (PNSE'07, PNSE'08, PNSE'09 and PNSE'10) made it into ToPNoC volumes in the Springer LNCS series (volumes 5100, 5460 and 5800).

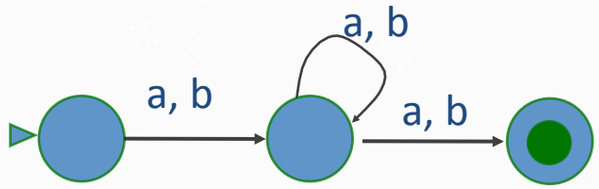
Chairs

- Michael Duvigneau (University of Hamburg, Germany)
- Lawrence Cabac (University of Hamburg, Germany)
- Daniel Moldt (University of Hamburg, Germany)

Formalização do processo de modelagem

A formalização do processo de modelagem de qualquer sistema dinâmico (ou de qualquer processo similar, seja qual for a aplicação) pode ser feito e Redes de Petri.

Para sistemas discretos, um formalismo alternativo e também muito genérico é proposto pela Computação Teórica com a Teoria de Autômatos. Na verdade esta representação também se enqudra no paradigma estado transição e pode descrever uma série de processos computacionais, onde o mais importante é o parser.



CS154 Introduction to Automata and ComplexityTheory

Jeffrey D. Ullman, Stanford University

- Finite automata are finite collections of states with transition rules that take you from one state to another.
- Original application was sequential switching circuits, where the "state" was the settings of internal bits.
- Today, several kinds of software can be modeled by FA.

CS154 Introduction to Automata and ComplexityTheory

Jeffrey D. Ullman, Stanford University

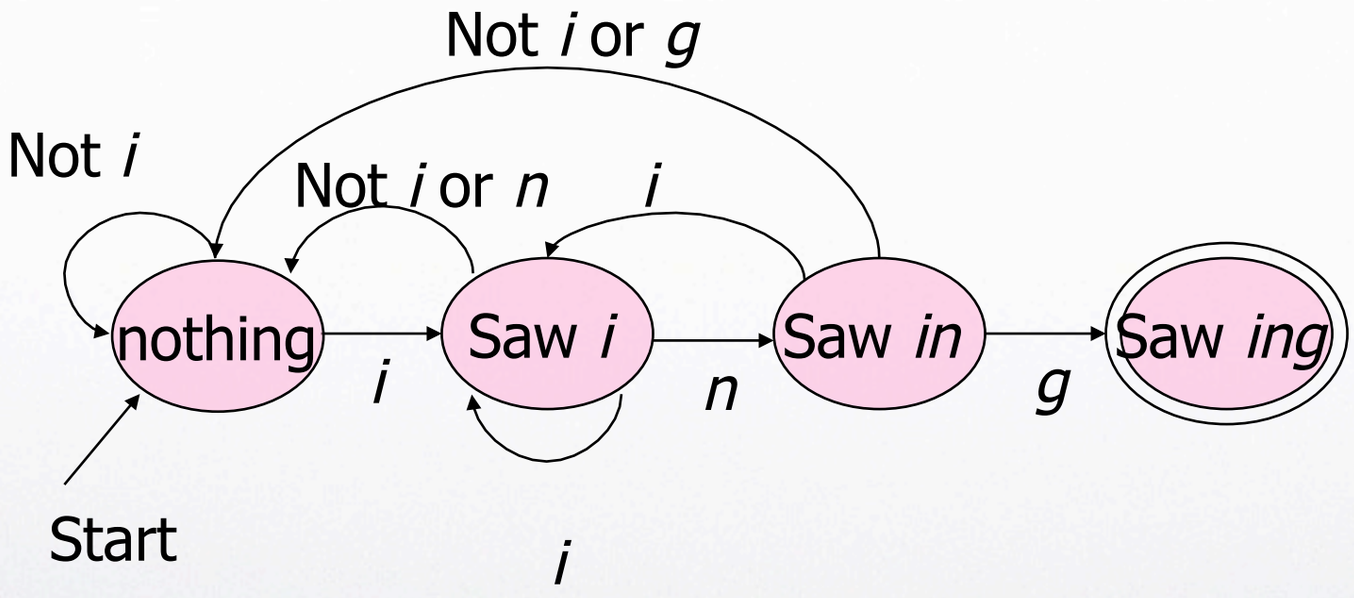
Representing FA

- Simplest representation is often a graph.
  - Nodes = states.
  - Arcs indicate state transitions.
  - Labels on arcs tell what causes the transition.

CS154 Introduction to Automata and Complexity Theory

Jeffrey D. Ullman, Stanford University

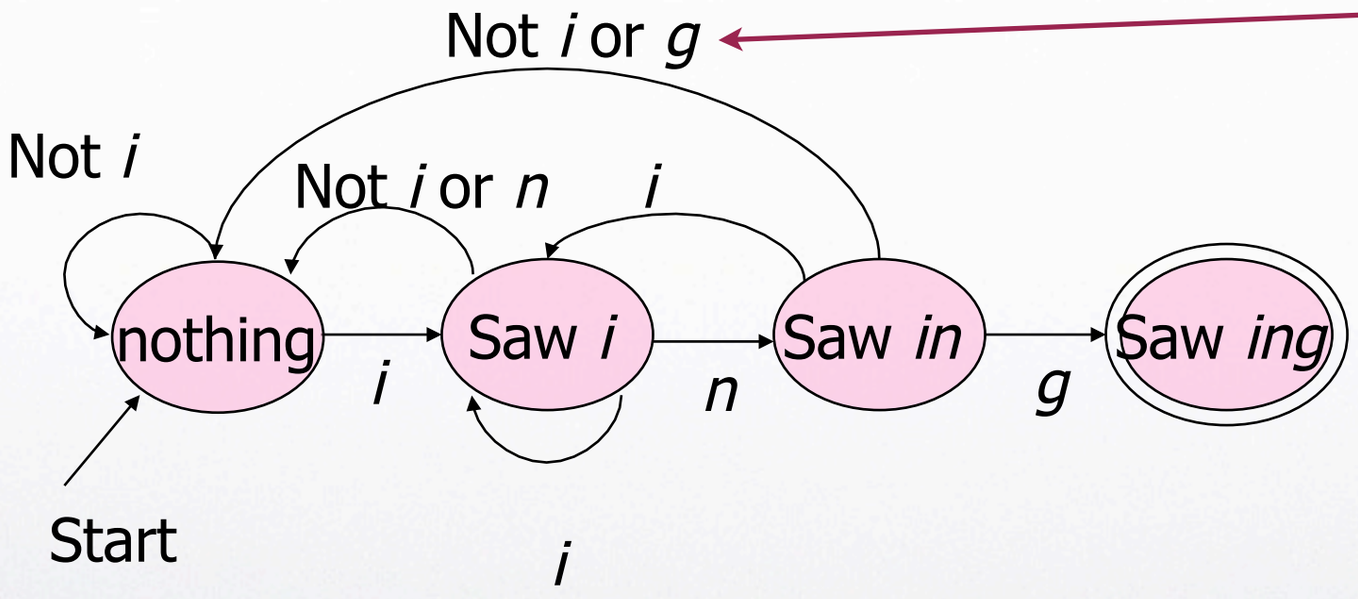
Example: Recognizing Strings Ending in "ing"



CS154 Introduction to Automata and Complexity Theory

Jeffrey D. Ullman, Stanford University

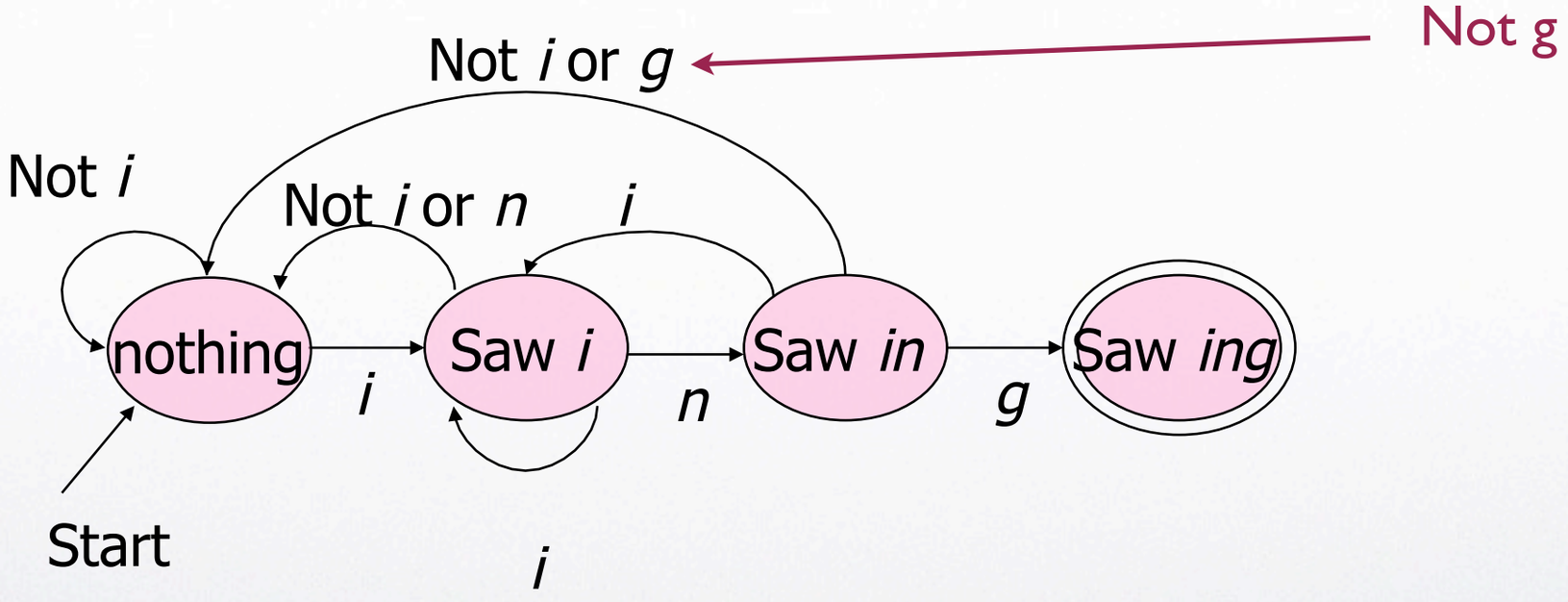
Example: Recognizing Strings Ending in "ing"



CS154 Introduction to Automata and Complexity Theory

Jeffrey D. Ullman, Stanford University

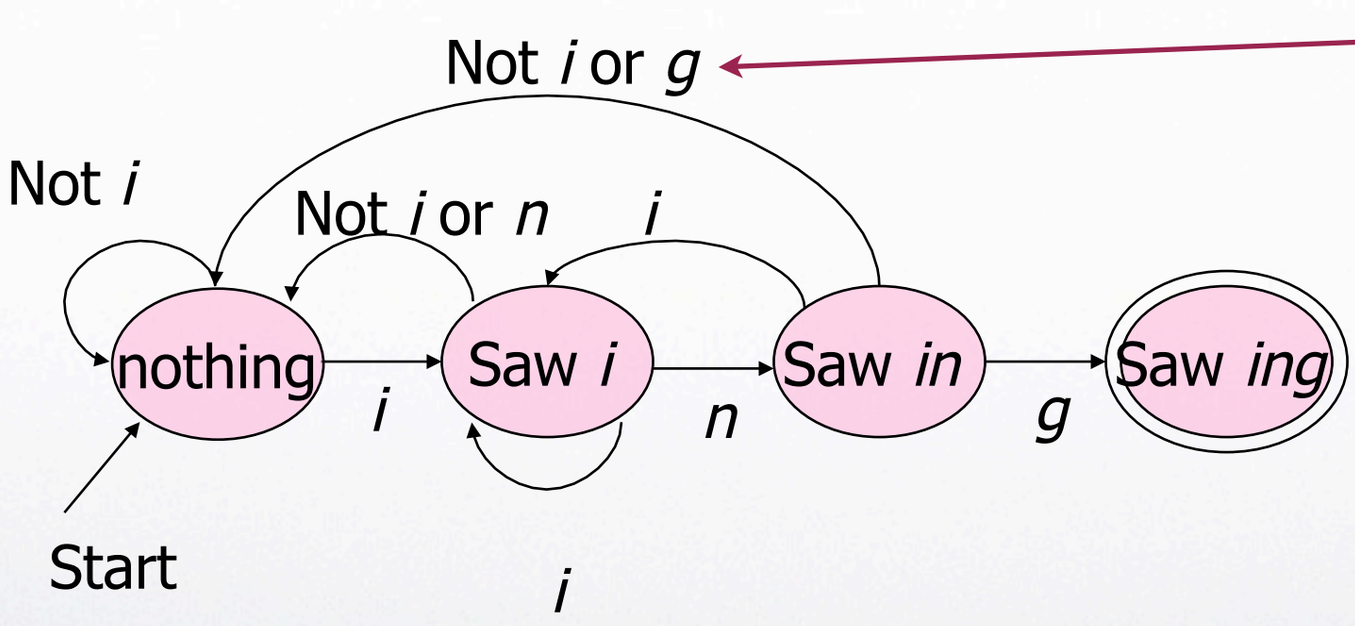
Example: Recognizing Strings Ending in "ing"



CS154 Introduction to Automata and Complexity Theory

Jeffrey D. Ullman, Stanford University

Example: Recognizing Strings Ending in "ing"



Experimente agora fazer o parsing da palavra "mining".

CS154 Introduction to Automata and Complexity Theory

Jeffrey D. Ullman, Stanford University

Deterministic Finite Automata

- A formalism for defining languages, consisting of:
  1. A finite set of *states* (Q, typically).
  2. An *input alphabet* (Σ, typically).
  3. A *transition function* (δ, typically).
  4. A *start state* (q_0, in Q, typically).
  5. A set of *final states* (F ⊆ Q, typically).
     - "Final" and "accepting" are synonyms.

1

CS154 Introduction to Automata and Complexity Theory

Jeffrey D. Ullman, Stanford University

The Transition Function

- Takes two arguments: a state and an input symbol.
- δ(q, a) = the state that the DFA goes to when it is in state *q* and input *a* is received.

7

CS154 Introduction to Automata and Complexity Theory

Jeffrey D. Ullman, Stanford University

Graph Representation of DFA's

- Nodes = states.
- Arcs represent transition function.
  - Arc from state p to state q labeled by all those input symbols that have transitions from p to q.
- Arrow labeled "Start" to the start state.
- Final states indicated by double circles.

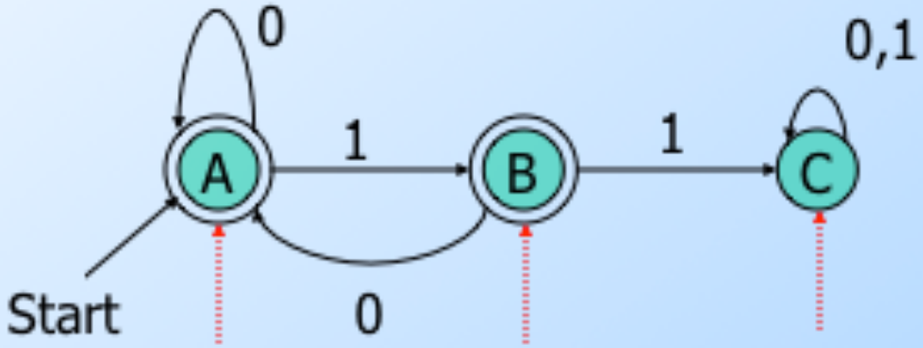
8

CS154 Introduction to Automata and ComplexityTheory

Jeffrey D. Ullman, Stanford University

Example: Graph of a DFA

Accepts all strings without two consecutive 1's.



Previous string OK, does not end in 1.

Previous String OK, ends in a single 1.

Consecutive 1's have been seen.

9

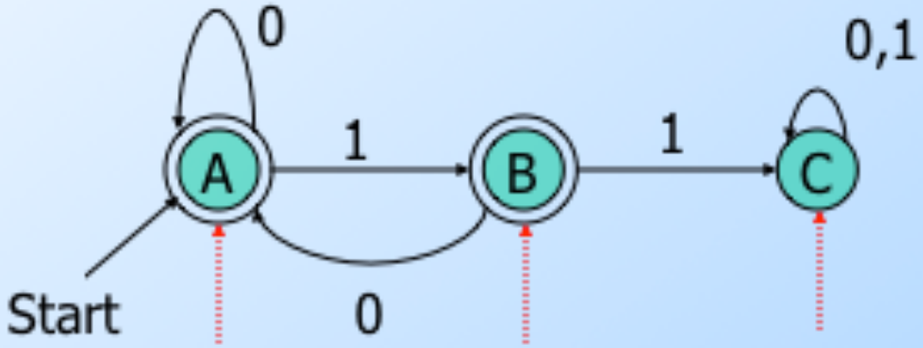
CS154 Introduction to Automata and Complexity Theory

Jeffrey D. Ullman, Stanford University

Também chamado Transition Graph

Example: Graph of a DFA

Accepts all strings without two consecutive 1's.



Previous string OK, does not end in 1.

Previous String OK, ends in a single 1.

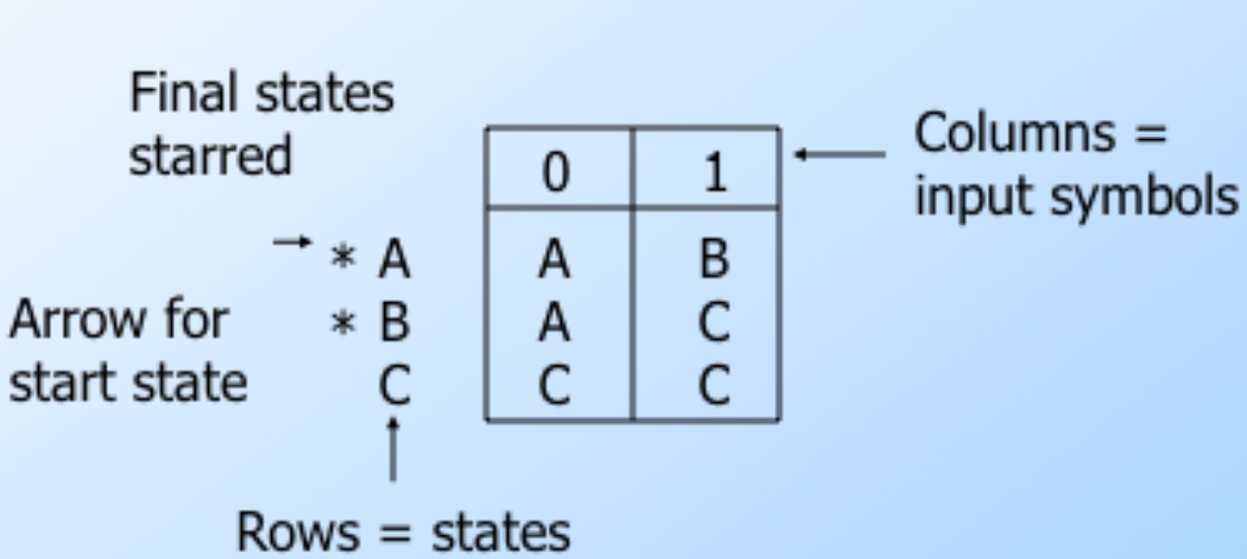
Consecutive 1's have been seen.

9

CS154 Introduction to Automata and Complexity Theory

Jeffrey D. Ullman, Stanford University

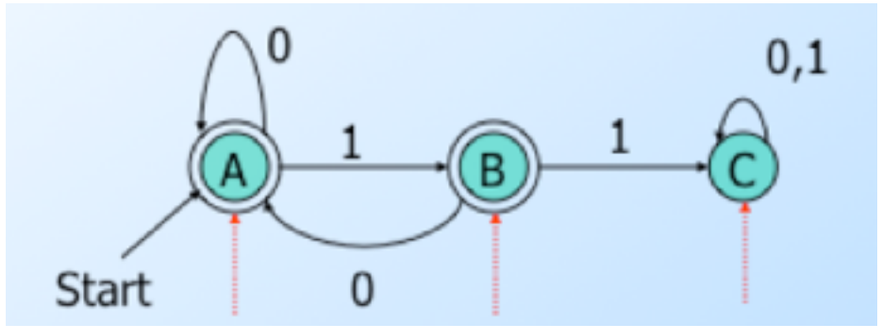
Alternative Representation: Transition Table

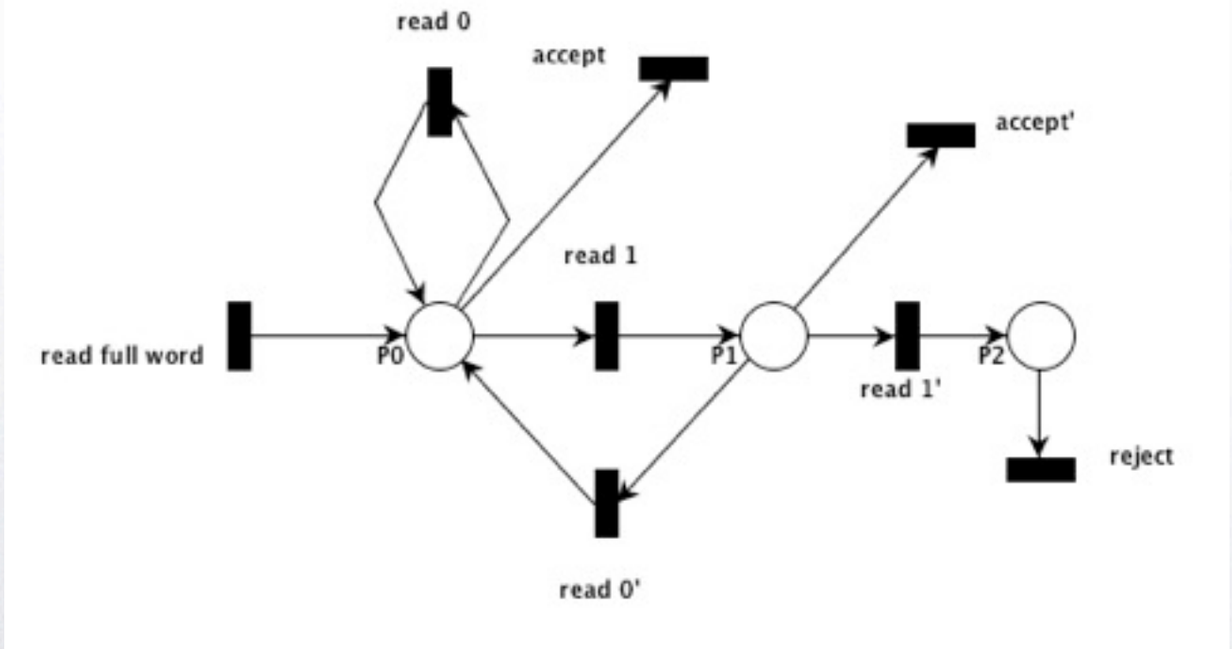


http://infolab.stanford.edu/~ullman/ialc/spr10/spr10.html#LECTURE%20NOTES

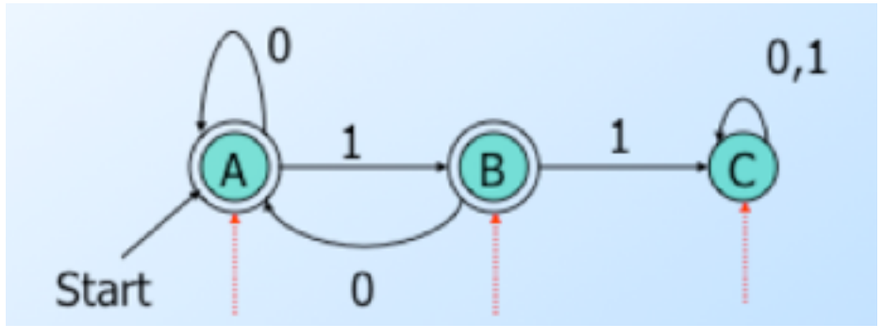
10

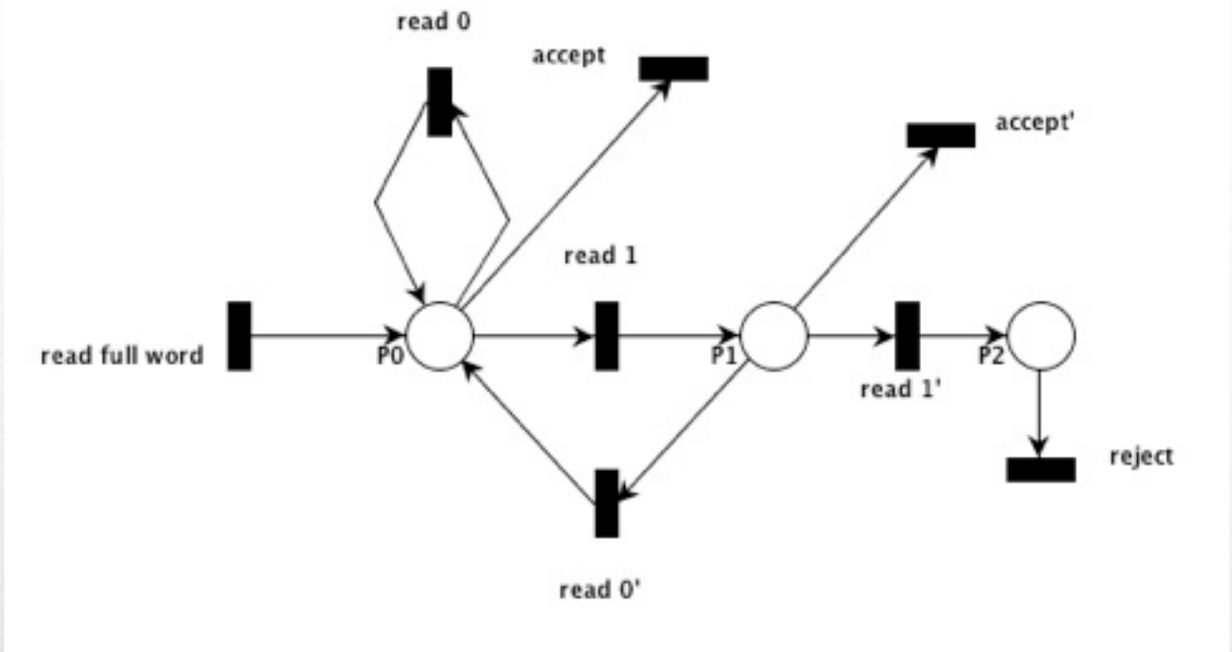
Note que um TG (transition graph) também pode ser representado por uma rede de Petri Elementar (que ainda não definimos). Por enquanto vamos suportar esta analogia mapeando estados do TG com lugares de uma RdP e arcos do TG com transições da RdP.





Note que um TG (transition graph) também pode ser representado por uma rede de Petri Elementar (que ainda não definimos). Por enquanto vamos suportar esta analogia mapeando estados do TG com lugares de uma RdP e arcos do TG com transições da RdP.





Analise o grafo acima e a rede à esquerda. Você se sentiria tentado a "reduzir" o número de elementos desta rede?

Modelagem Estado/Transição com Redes de Petri

- Estados e transições são noções distintas porém intercaladas;
- Ambos, estados e transições são entidades distribuidas;
- A extensão das mudanças causadas por uma transição é restrita e não depende do estado em que esta ocorre;
- Uma transição está *habilitada* em um estado sse as mudanças associadas à transição podem ocorrer neste estado, na extensão prefixada anteriormente.

Estados e transições

O comportamento de um sistema dinâmico é representado por um estado S distribuído, formado por um conjunto de estados atômicos, como os que foram representados no exemplo simples do parser mostrado anteriormente.

Similarmente, as transições serão representadas por um conjunto de transições atômicas T desde que estejam todas habilitadas simultâneamente, de modo que estes conjuntos satisfazem à relação,

Disjunção entre estados e transições

- $S \cap T = \phi$.

Portanto estados e transições são ambos elementos distribuídos:

Um estado distribuído é dado por um conjunto de condições válidas simultâneamente, isto é,

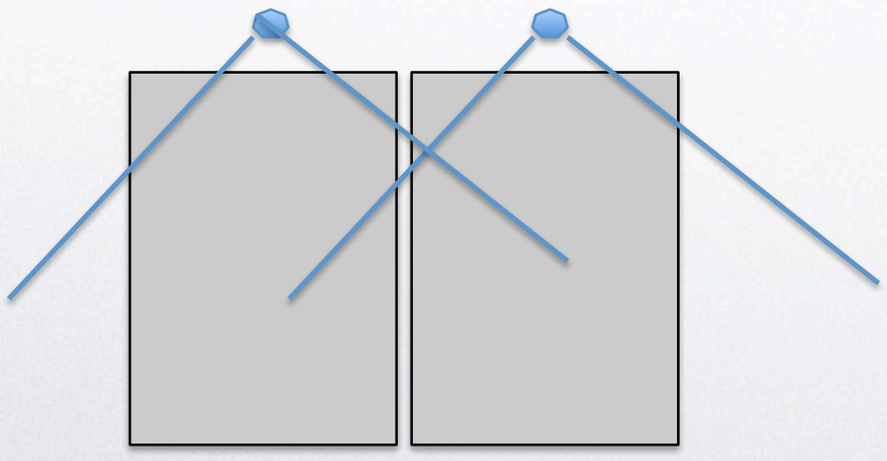
- $\{s_1, s_2, \ldots, s_n\} \Longrightarrow case.$

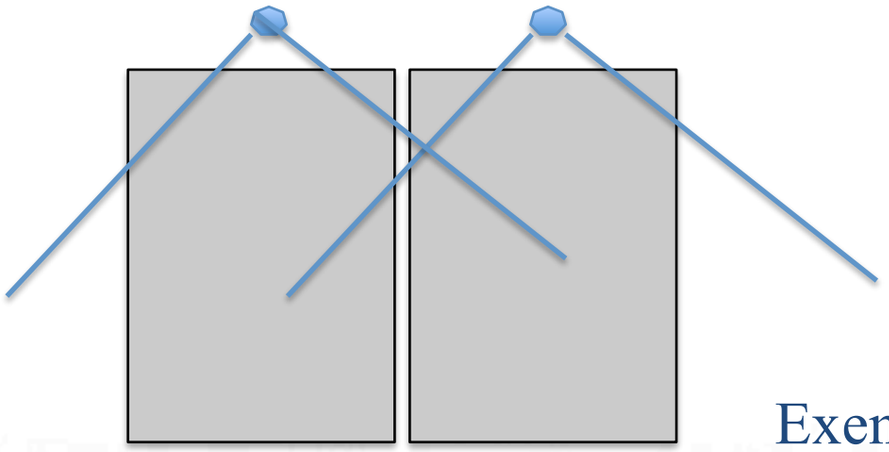
Uma transição distribuída é dada por um conjunto de transições válidas simultâneamente, isto é,

- $\{t_1, t_2, \ldots, t_m\} \Longrightarrow passo.$

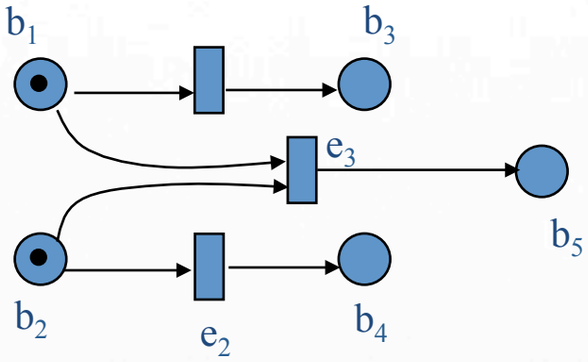
Um exemplo simples do que estamos falando, aplicado a sistemas de automação é a representação do comportamento de uma porta automática, que aciona motores para abertura e fechamento dependendo de sinal de sensores.

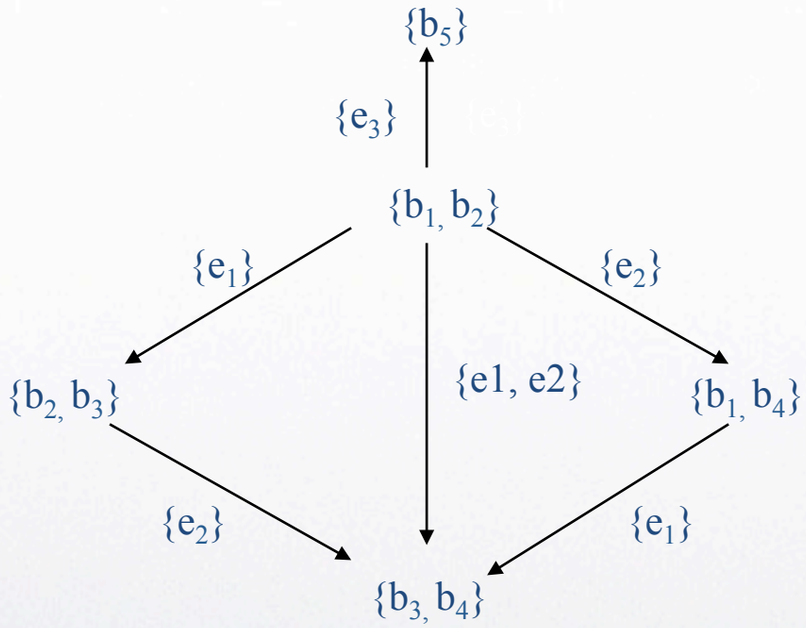
Um modelo bastante simples (e dificilmente encontrado na prática) é dado pelo esquema abaixo, onde cada porta te um sensor com varredura determinada. Ao detectar a aproximação de uma pessoa a porta correspondente se abre, ou os dois lados se abrem no caso das duas detectarem simultâneamente a aproximação.





Exemplo (Thiagarajan) :





B = { b_1, b_2, b_3, b_4, b_5 }

E = { e_1, e_2, e_3}

C = {{b_1, b_2}, {{b_2, b_3}, {{b_1, b_4}, {{b_3, b_4}, {{b_5}}

U = {{e_1}, {e_2}, {e_1, e_2}, {e_3} }

Grafo de atingibilidade

Definição de Rede de Petri

Definition

Definition 1] Uma rede de Petri é um grafo direcionado, simples, bipartido e conexo, representado pela n-upla $N = (S, T; F)$, onde S é um conjunto de estados $\{s_i\}$, T é um conjunto de transições $\{t_j\}$, e F é uma relação de transição (o relação de fluxo), tal que:

i) $S \cap T = \emptyset$ e $S \cup T \neq \emptyset$;

ii) $F \subseteq (S \times T) \cup (T \times S)$;

iii) $dom(F) \cup ran(F) = S \cup T$, onde

$dom(F) = \{x \in (S \cup T) \mid \exists y \in (S \cup T).(s, y) \in F\}$,

$ran(F) = \{y \in (S \cup T) \mid \exists x \in (S \cup T).(x, y) \in F\}$.

Princípios para modelagem em Redes de Petri

As redes possuem propriedades típicas dos esquemas que as tornam
Uma excelente representação formal para sistemas (dinâmicos) discretos,
Entre os quais figuram :

- o princípio da dualidade
- o princípio da localidade
- o princípio da concorrência
- o principio da representação gráfica
- o princípio da representação algébrica

O princípio da Dualidade

Existem dois tipos de elementos na rede: os elementos ditos passivos ou P-elementos (S-elementos), que representam estados, e os elementos ditos ativos ou T-elementos que representam mudança de estado ou transição.

Estes elementos são disjuntos, isto é, não existe na natureza nada que possa Possuir ambas as propriedades e questione que $P \cap T = \varnothing$

Exemplo: a largada na fórmula 1

Carro A

Carro B

Girault, C. & Valk, R.; Petri Nets for Systejms Engineering, Springer, 2003

Identificando os Estados

p_1= carro A: preparando-se para começar;
p_2= carro A: esperando o sinal de largada;
p_3= carro A: correndo;
p_4= sinal de prontidão do carro A enviado;
p_5= sinal de largada para o carro A enviado;
p_6= operador: esperando sinal de prontidão dos pilotos;
p_7= operador: sinal de largada enviado;
p_8= sinal de prontidão do carro B enviado;
p_9= sinal de largada para o carro B enviado;
p_{10}= carro B: preparando-se para começar;
p_{11}= carro B: esperando o sinal de largada;
p_{12}= carro B: correndo;

Identificando as transições

t_1 = carro A: envia sinal de prontidão
t_2 = carro A: acelera
t_3 = operador: manda sinal de largada
t_4 = carro B: envia sinal de prontidão
t_5 = carro B: acelera

O estado inicial

O estado inicial (valor verdade das condições que compõem o estado):

M_1= [p1=T, p2=F, p3=F, p4=F, p5=F, p6=T, p7=F, p8=F, p9=F, p10=T, p11=F, p12=F]

p_1= carro A: preparando-se para começar;
p_2= carro A: esperando o sinal de largada;
p_3= carro A: correndo;
p_4= sinal de prontidão do carro A enviado;
p_5= sinal de largada para o carro A enviado;
p_6= operador: esperando sinal de prontidão dos pilotos;
p_7= operador: sinal de largada enviado;
p_8= sinal de prontidão do carro B enviado;
p_9= sinal de largada para o carro B enviado;
p_{10}= carro B: preparando-se para começar;
p_{11}= carro B: esperando o sinal de largada;
p_{12}= carro B: correndo;

A dinâmica Estado/ Transição

M_1= [p1=T, p2=F, p3=F, p4=F, p5=F, p6=T, p7=F, p8=F, p9=F, p10=T, p11=F, p12=F]

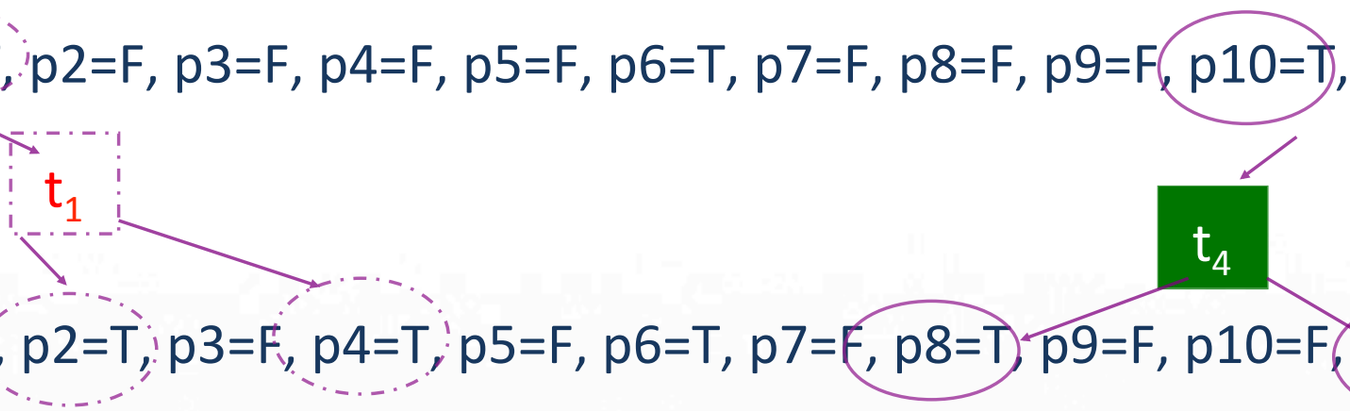
M_2= [p1=F, p2=V, p3=F, p4=T, p5=F, p6=T, p7=F, p8=F, p9=F, p10=T, p11=F, p12=F]

p_1= carro A: preparando-se para começar;
p_2= carro A: esperando o sinal de largada;
p_3= carro A: correndo;
p_4= sinal de prontidão do carro A enviado;
p_5= sinal de largada para o carro A enviado;
p_6= operador: esperando sinal de prontidão dos pilotos;
p_7= operador: sinal de largada enviado;
p_8= sinal de prontidão do carro B enviado;
p_9= sinal de largada para o carro B enviado;
p_{10}= carro B: preparando-se para começar;
p_{11}= carro B: esperando o sinal de largada;
p_{12}= carro B: correndo;

t_1 = carro A: envia sinal de prontidão
t_2 = carro A: acelera
t_3 = operador: manda sinal de largada
t_4 = carro B: envia sinal de prontidão
t_5 = carro B: acelera

Transições independentes

M_1= [p1=T, p2=F, p3=F, p4=F, p5=F, p6=T, p7=F, p8=F, p9=F, p10=T, p11=F, p12=F]



M_2= [p1=F, p2=T, p3=F, p4=T, p5=F, p6=T, p7=F, p8=T, p9=F, p10=F, p11=T, p12=F]

p_1= carro A: preparando-se para começar;
p_2= carro A: esperando o sinal de largada;
p_3= carro A: correndo;
p_4= sinal de prontidão do carro A enviado;
p_5= sinal de largada para o carro A enviado;
p_6= operador: esperando sinal de prontidão dos pilotos;
p_7= operador: sinal de largada enviado;
p_8= sinal de prontidão do carro B enviado;
p_9= sinal de largada para o carro B enviado;
p_{10}= carro B: preparando-se para começar;
p_{11}= carro B: esperando o sinal de largada;
p_{12}= carro B: correndo;

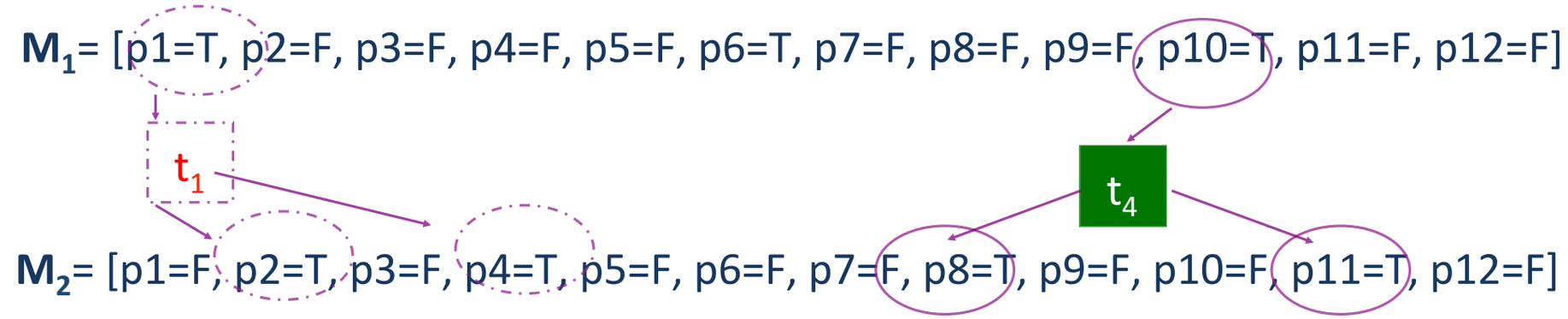
t_1 = carro A: envia sinal de prontidão
t_2 = carro A: acelera
t_3 = operador: manda sinal de largada
t_4 = carro B: envia sinal de prontidão
t_5 = carro B: acelera

O princípio da localidade

O pricípio da localidade é atribuído a transições

A localidade de uma transição é dada pelo conjunto das suas pré-condições unido ao cojunto das pós-condições

Pelo princípio da localidade o comportamento das transições depende unicamente da sua localidade.



p_1= carro A: preparando-se para começar;
p_2= carro A: esperando o sinal de largada;
p_3= carro A: correndo;
p_4= sinal de prontidão do carro A enviado;
p_5= sinal de largada para o carro A enviado;
p_6= operador: esperando sinal de prontidão dos pilotos;
p_7= operador: sinal de largada enviado;
p_8= sinal de prontidão do carro B enviado;
p_9= sinal de largada para o carro B enviado;
p_{10}= carro B: preparando-se para começar;
p_{11}= carro B: esperando o sinal de largada;
p_{12}= carro B: correndo;

t_1 = carro A: envia sinal de prontidão
t_2 = carro A: acelera
t_3 = operador: manda sinal de largada
t_4 = carro B: envia sinal de prontidão
t_5 = carro B: acelera

Eventos independentes

M_1= [p1=T, p2=F, p3=F, p4=F, p5=F, p6=T, p7=F, p8=F, p9=F, p10=T, p11=F, p12=F]

M_3= [p1=F, p2=T, p3=F, p4=T, p5=F, p6=T, p7=F, p8=T, p9=F, p10=F, p11=T, p12=F]

p_1= carro A: preparando-se para começar;
p_2= carro A: esperando o sinal de largada;
p_3= carro A: correndo;
p_4= sinal de prontidão do carro A enviado;
p_5= sinal de largada para o carro A enviado;
p_6= operador: esperando sinal de prontidão dos pilotos;
p_7= operador: sinal de largada enviado;
p_8= sinal de prontidão do carro B enviado;
p_9= sinal de largada para o carro B enviado;
p_{10}= carro B: preparando-se para começar;
p_{11}= carro B: esperando o sinal de largada;
p_{12}= carro B: correndo;

t_1 = carro A: envia sinal de prontidão
t_2 = carro A: acelera
t_3 = operador: manda sinal de largada
t_4 = carro B: envia sinal de prontidão
t_5 = carro B: acelera

Princípio da concorrência

Princípio da Concorrência

- Duas transições t_1 e t_2 são cocorrentes se e somente se suas locaidades são disjuntas, isto é,

$$Indep(t_1, t_2) \Leftrightarrow \{Loc(t_1) \cap Loc(t_2) = \oslash\}$$

Princípio da Representação Gráfica

Qualquer sistema representado em Redes de Petri admite uma representação gráfica, e esta representação obedece a regras rígidas que dão consistência ao modelo.

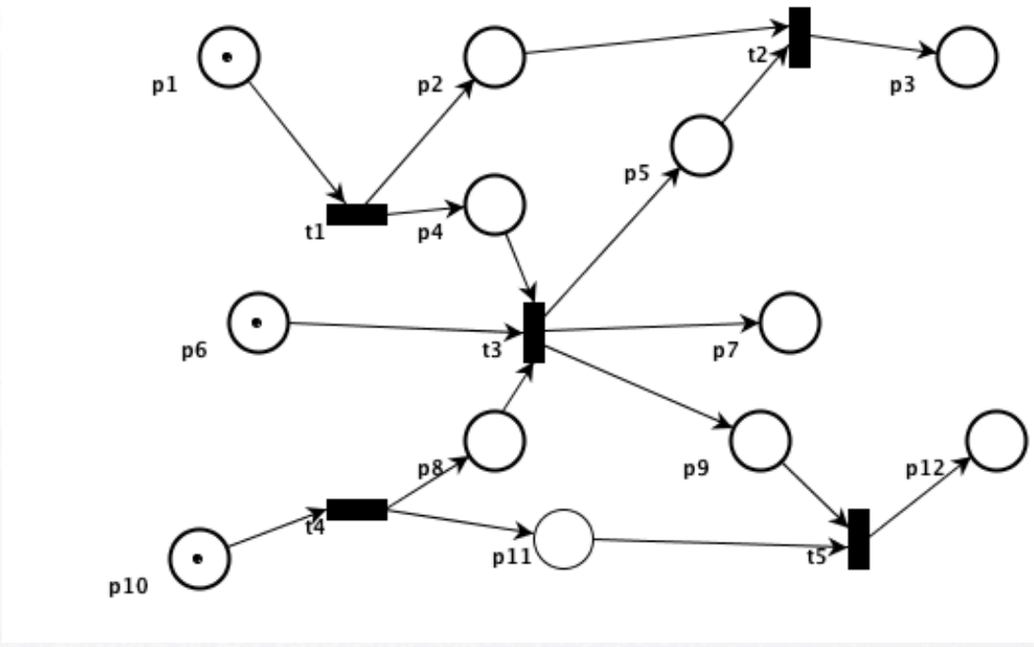
Elementos para representação gráfica

- Os elementos ditos passivos (por convenção) são chamados lugares (ou S-elementos, do termo sahlen em alemão) e são representados graficamente por circulos ou elipses.
- Os elementos ditos ativos (por convenção) são chamados transições e são representados por retângulos ou barras.
- Lugares e trasições são ligados por arcos orientados de modo que um arco liga sempres um lugar a uma transição ou vice-versa, mas nunca dois lugares ou duas transições.

Modelagem Estado/Transição
X
Modelagem em Redes de Petri

- Estados e transições são noções distintas porém intercaladas;
- Ambos, estados e transições são entidades distribuidas;
- A extensão das mudanças causadas por uma transição é restrita e não depende do estado em que esta ocorre;
- Uma transição está *habilitada* em um estado sse as mudanças associadas à transição podem ocorrer neste estado, na extensão prefixada anteriormente.

Voltando ao exemplo



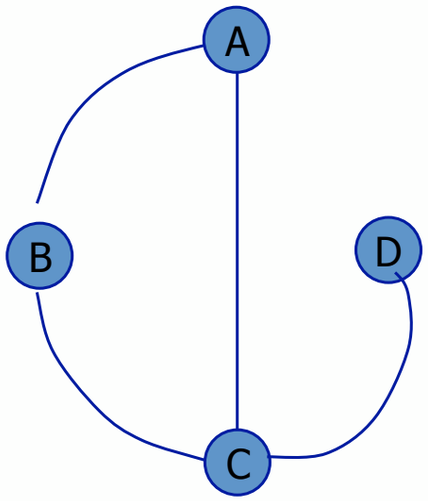
p_1= carro A: preparando-se para começar;
p_2= carro A: esperando o sinal de largada;
p_3= carro A: correndo;
p_4= sinal de prontidão do carro A enviado;
p_5= sinal de largada para o carro A enviado;
p_6= operador: esperando sinal de prontidão dos pilotos;
p_7= operador: sinal de largada enviado;
p_8= sinal de prontidão do carro B enviado;
p_9= sinal de largada para o carro B enviado;
p_{10}= carro B: preparando-se para começar;
p_{11}= carro B: esperando o sinal de largada;
p_{12}= carro B: correndo;

t_1 = carro A: envia sinal de prontidão
t_2 = carro A: acelera
t_3 = operador: manda sinal de largada
t_4 = carro B: envia sinal de prontidão
t_5 = carro B: acelera

Princípio da Representação Algébrica

Todo sistema modelado em Redes de Petri admite uma e somente uma representação algébrica, que é dada pelas matrizes que representam a estrutura do modelo (estados e transições distribuídas) e por uma equação de estado que representa a evolução dos estados pela ocorrência de transições habilitadas.

A relação entre representação gráfica e matrizes é dada pela teoria de grafos, onde um grafo simples não-orientado pode ser representado por uma matriz.



	A	B	C	D
A	0	1	1	0
B	1	0	1	0
C	1	1	0	1
D	0	0	1	0

Matriz de Incidência

No caso dos grafos todos os nós podem ser conectados por um arco. Em princípio, portanto a matriz mapeia as vizinhaças (a existência de arcos) entre os nós e de fato é uma tabela de dupla entrada. Para os pares de V X V que sejam vizinhos a posição respectiva recebe um "1"; se não existe vizinhança a posição recebe "0".

Naturalmente, para garfos simples, a diagonal principal é nula, e, se o grafo é não-direcionado, a matriz é simétrica com relação a diagonal principal.

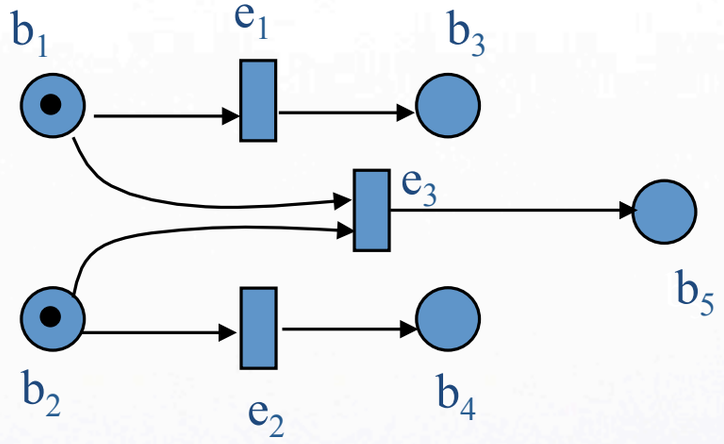
No caso das redes de Petri temos que:

i) se a rede é elementar a matriz de incidência será também unimodular como no caso dos grafos, mas para outros tipos de rede poderemos ter mais de um arco entre lugares e transições e os elementos da matriz poderão assumir valores maiores que a unidade;

ii) as redes de Petri são um grafo bi-partido, isto é, um grafo onde o conjunto de vértices V é dividido em dois conjuntos (o dos lugares e o das transições) e só pode haver vizinhança entre elementos destes dois conjuntos. Assim, se o número de transições e lugares não coincide a matriz não é necessáriamente quadrada e não necessariamente tem um determinante.

Exemplo:

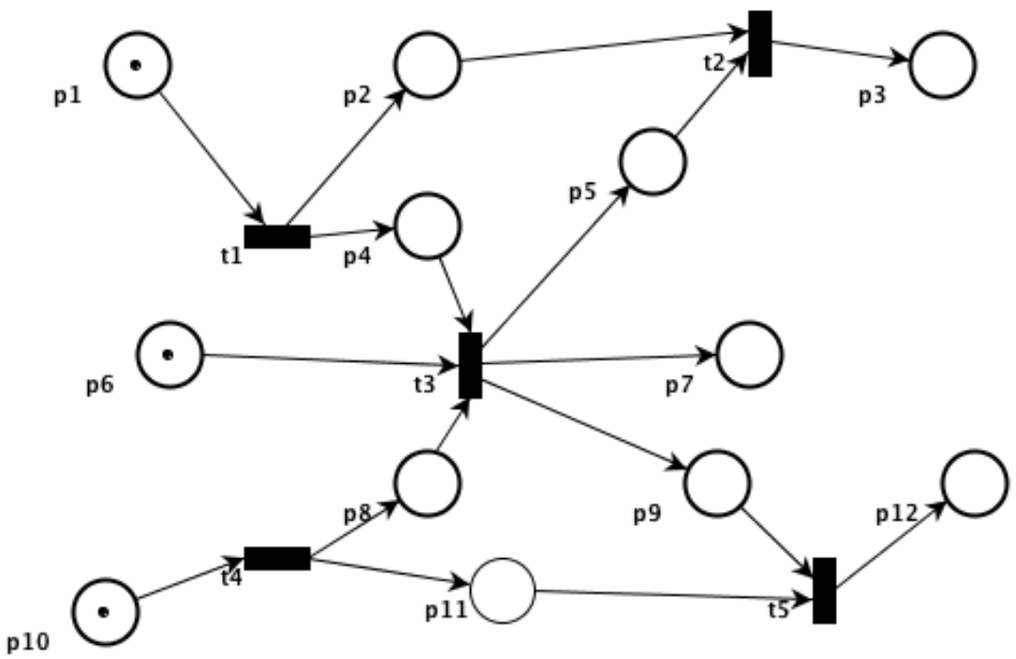
Usando o mesmo exemplo da porta automática podemos construir a matriz de incidência desta rede.

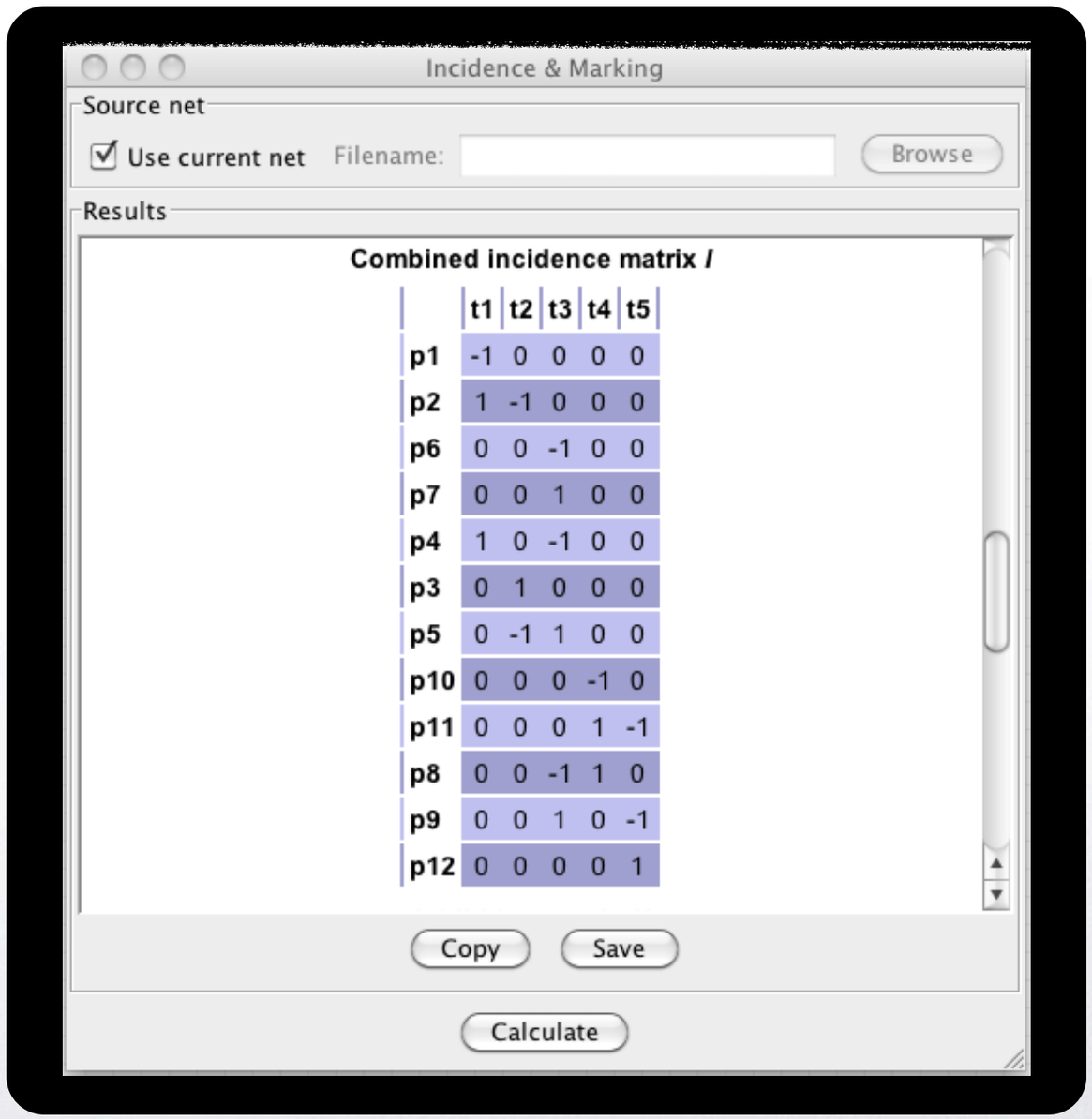


No caso das redes se faz uma tabela de dupla entrada das transições contra os lugares. Cada elemento da matriz indica se o lugar é incidente, -1, emergente, 1, ou se não há conexão de nenhum tipo, 0.

$$\begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \end{array} \begin{array}{c} \begin{array}{ccccc} b_1 & b_2 & b_3 & b_4 & b_5 \end{array} \\ \begin{bmatrix} -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ -1 & -1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Exemplo da largada de Fórmula 1





Redes de Petri: Definição

Definition

Definition 1] Uma rede de Petri é um grafo direcionado, simples, bipartido e conexo, representado pela n-upla $N = (S, T; F)$, onde S é um conjunto de estados $\{s_i\}$, T é um conjunto de transições $\{t_j\}$, e F é uma relação de transição (o relação de fluxo), tal que:

i) $S \cap T = \emptyset$ e $S \cup T \neq \emptyset$;

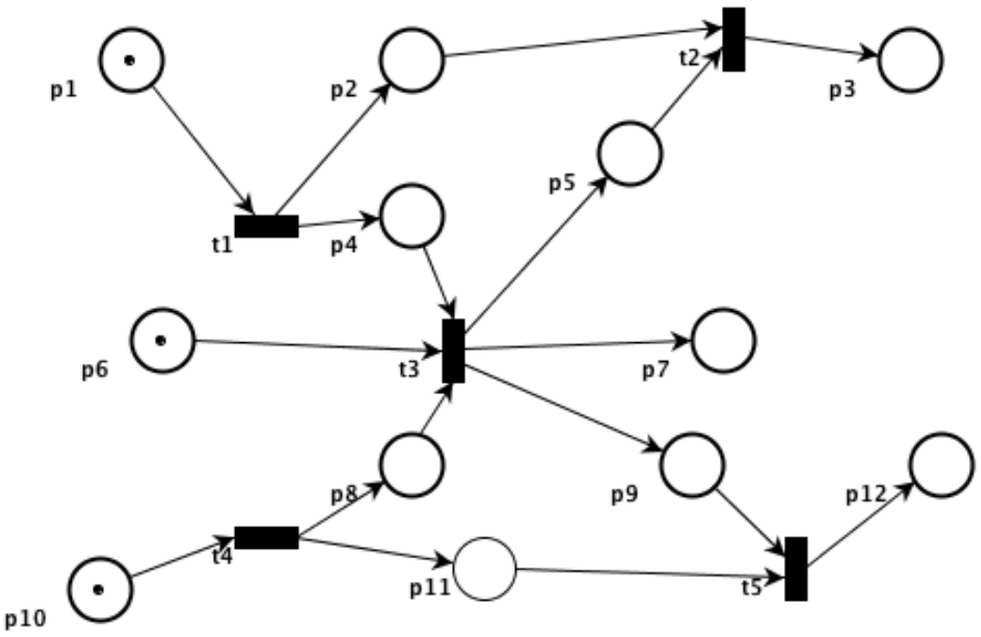
ii) $F \subseteq (S \times T) \cup (T \times S)$;

iii) $dom(F) \cup ran(F) = S \cup T$, onde

$dom(F) = \{x \in (S \cup T) \mid \exists y \in (S \cup T).(s, y) \in F\}$,

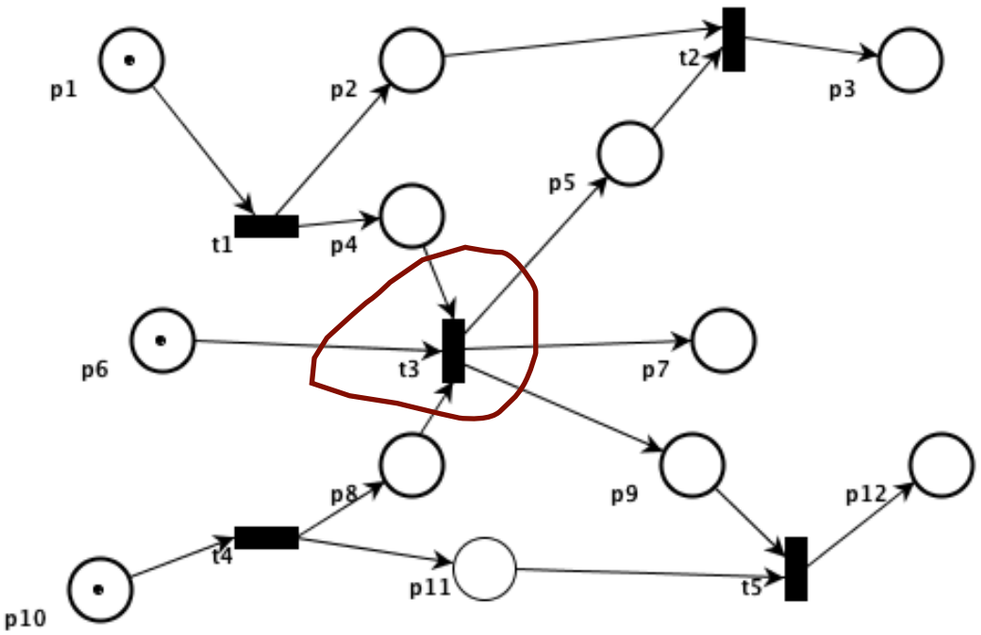
$ran(F) = \{y \in (S \cup T) \mid \exists x \in (S \cup T).(x, y) \in F\}$.

Configurações especiais: sincronismo



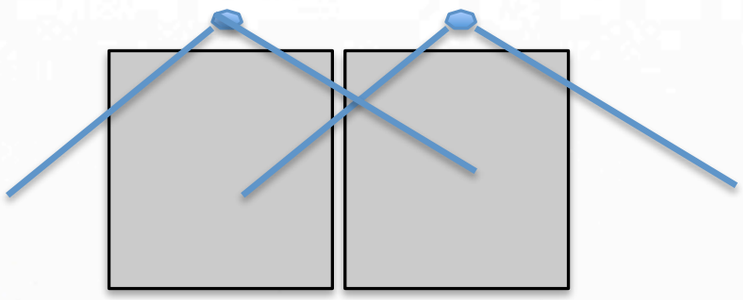
A convergência de diversos caminhos na rede para uma única transição sincroniza as ações, uma vez que a evolução dos estados por esta transição só ocorre quando todos os lugares que antecedem a transição estão marcados.

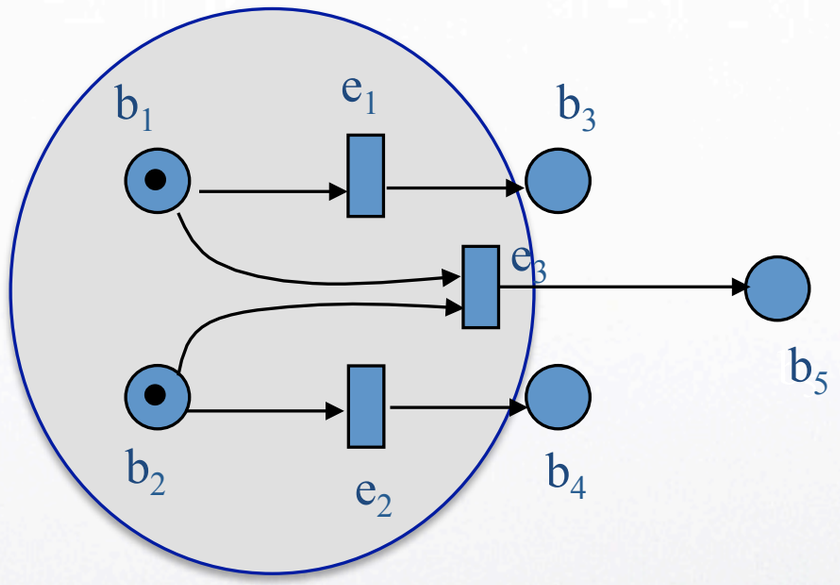
Configurações especiais: sincronismo



A convergência de diversos caminhos na rede para uma única transição sincroniza as ações, uma vez que a evolução dos estados por esta transição só ocorre quando todos os lugares que antecedem a transição estão marcados.

Configurações especiais: conflito

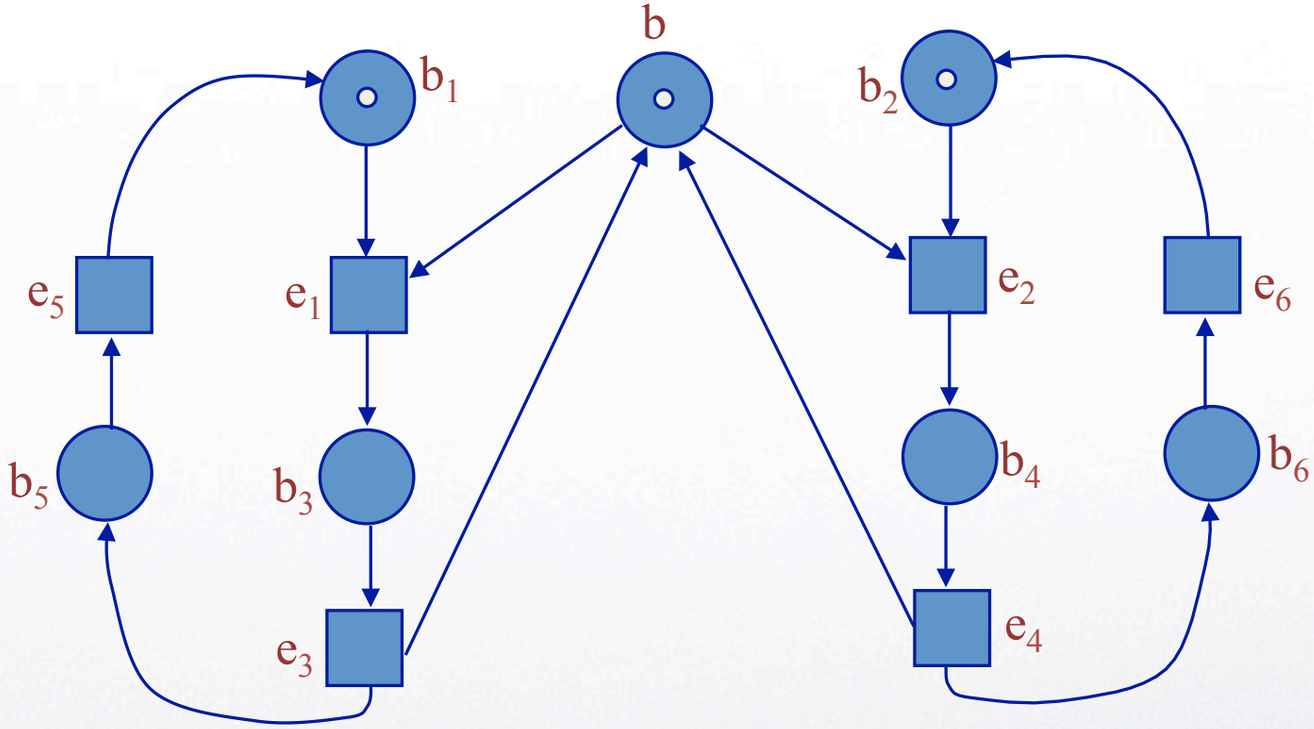




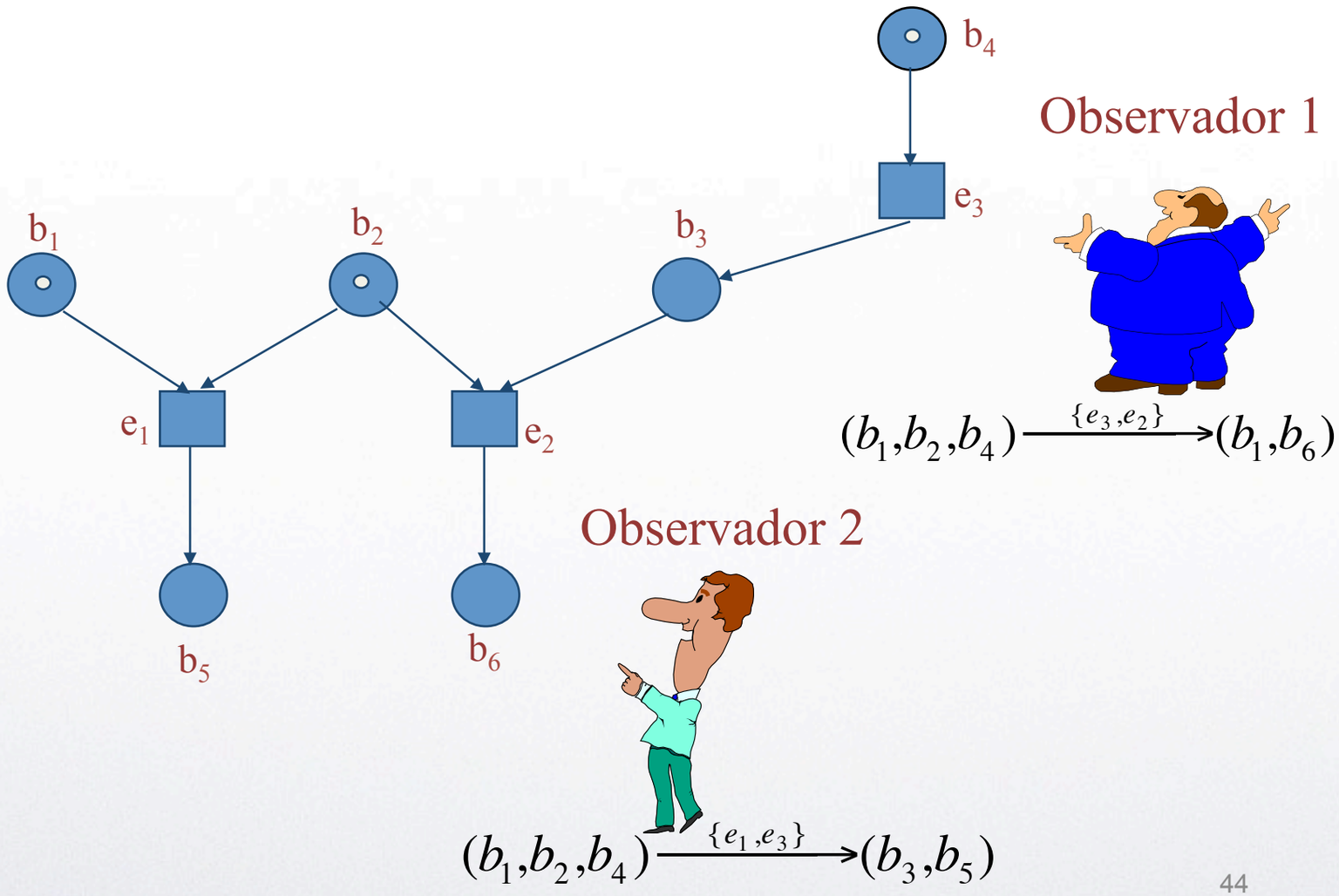
O Conflito denota uma situação de exclusão mútua (mutex), onde a ocorrência de uma transição inibe outras. No caso do exemplo as transições {e1, e2} são mutuamente exclusivas à transição {e3}.

Semáforo: o mutex clássico

Em uma situação genérica a mútua exclusão (mutex) é caracterizada por processos Itransições independentes que compartilham pelo menos um lugar. O inicio de um processo desabilita este lugar e faz com que o outro seja desativado.



Configurações especiais: contato ou confusão



Definindo pre-set e pos-set

Para formalizar a definição de localidade precisamos do conceito de pre-set e pos-set de um elemento da rede (lugar ou transição):

Def. 2

Seja uma rede de Petri $N = (S, T; F)$e sejam $X_N = S \cup T$, $x \in X_N$ e $Y_N \subseteq X_N$:

- $\bullet x = \{y \in X_N \mid (y, x) \in F\}$
- $x \bullet = \{y \in X_N \mid (x.y) \in F\}$
- $\bullet Y = \bigcup_{(x \in Y)} \bullet x$ e $Y \bullet = \bigcup_{(x \in Y)} x \bullet$

Definindo localidade

A localidade de um elemento da rede pode ser definida do seguinte modo:

Def. 3

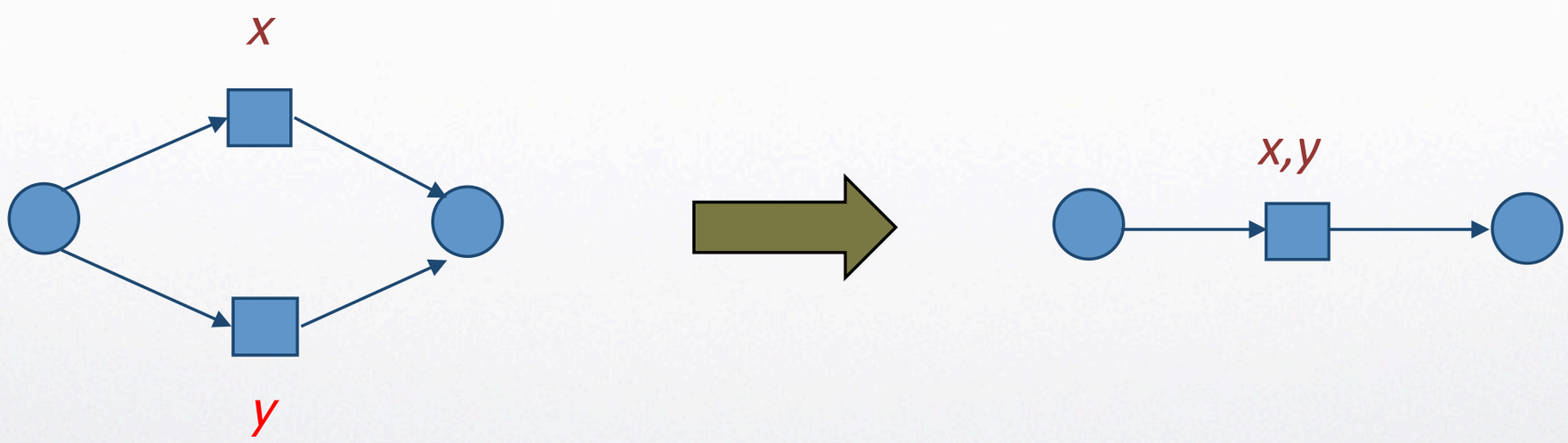
Seja uma rede de Petri $N = (S, T; F)$e seja $X_N = S \cup T$, $x \in X_N$. A localidade de um elemento genérico x é definida como:
$Loc(x) = \{x\} \cup \bullet x \cup x \bullet$

A convenção de considerar sempre as transições como elemento ativo nos leva a considerar de forma privilegiada a localidade das transições como elemento de análise.

Def. 4

Seja uma rede de Petri $N = (S, T; F)$ e sejam $X_N = S \cup T$, $x \in X_N$.

- N é dita uma rede pura se e somente se $\forall x \in X_N \bullet x \cap x \bullet = \oslash$;
- N é dita uma rede simples se e somente se $\forall x, y \in X_N [(\bullet x = \bullet y) \wedge (x \bullet = y \bullet) \Longrightarrow (x = y)]$.



Estados distribuídos

Def. 5

Seja uma rede de Petri $N = (S, T; F)$. Um subconjunto $M \subseteq S$ de condições válidas em um dado instante é dito um estado (distribuído) do sistema modelado por N.

Um estado também pode ser representado por um vetor

$$\vec{M} = \begin{pmatrix} s_1 \\ \vdots \\ s_m \end{pmatrix} \text{ onde } s_i = 1 \text{ se o } s_i \in M.$$

Transições elementares

Def. 6

Seja uma rede de Petri $N = (S, T; F)$. Uma transição elementar $t \in T$ está habilitada em um estado $M \subseteq S$ se e somente se $(\bullet t \subseteq M) \wedge (t \bullet \cap M = \oslash)$.

Esta definição insere uma regra de habilitação estrita. Mais tarde poderemos flexibilizar esta regra considerando somente o primeiro termo da conjunção.

Evolução Estado/Transição

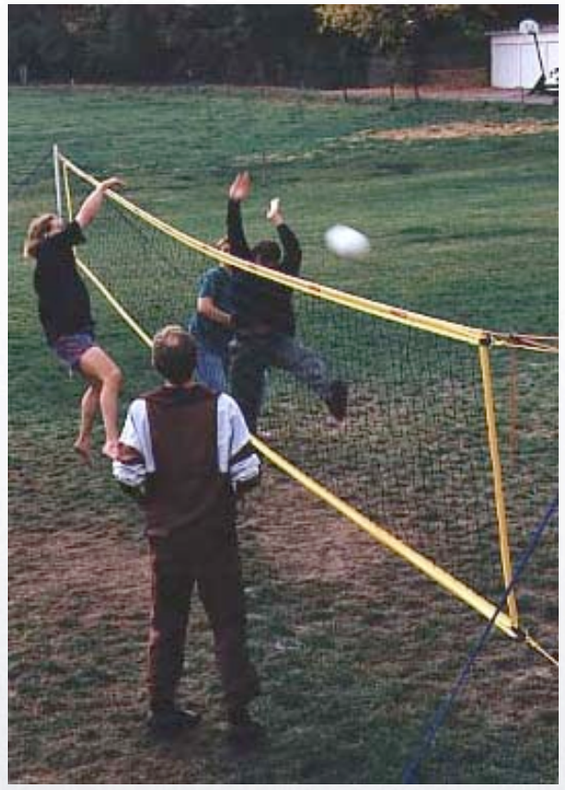
Def. 7

Seja uma rede de Petri $N = (S, T; F)$. Se uma transição elementar $t \in T$ habilitada em um estado $M \subseteq S$ ocorre, o estado M evului para o estado $M' = (M \setminus {\bullet t}) \cup t{\bullet}$.

Até aqui estamos nos restringindo à rede mais intuitiva, isto é, as redes elementares, onde cada lugar admite no máximo uma marca. Embora seja conveniente para processos lógicos e sequenciais e circuitos digitais, esta restrição é muito forte e será removida mais adiante para ter uma rede mais geral, chamada Place/Transition ou simplesmente P/T.

Informal Introduction to Petri Nets, W.Reisig, G. Rozemberg, LNCS 1491, 1-10.

3rd. Advanced Course in Petri Nets, Dagstuhl, Germany, October, 1996



PMR 5237

1o. ciclo de 2013

12 aulas
5 listas
4 milestones
1 artigo
1 prova

L - média das listas de exercício

nl - número de listas feitas

nt - número total de listas (5)

Tf - média do trabalho final (artigo)

nm - número de milestones feitos

nr - número total de milestones

P - Nota da prova

$$Mf = (4.(nl/nt)L + 4.(nm/nr)Tf + 2.P)/10$$

Stoa - USP

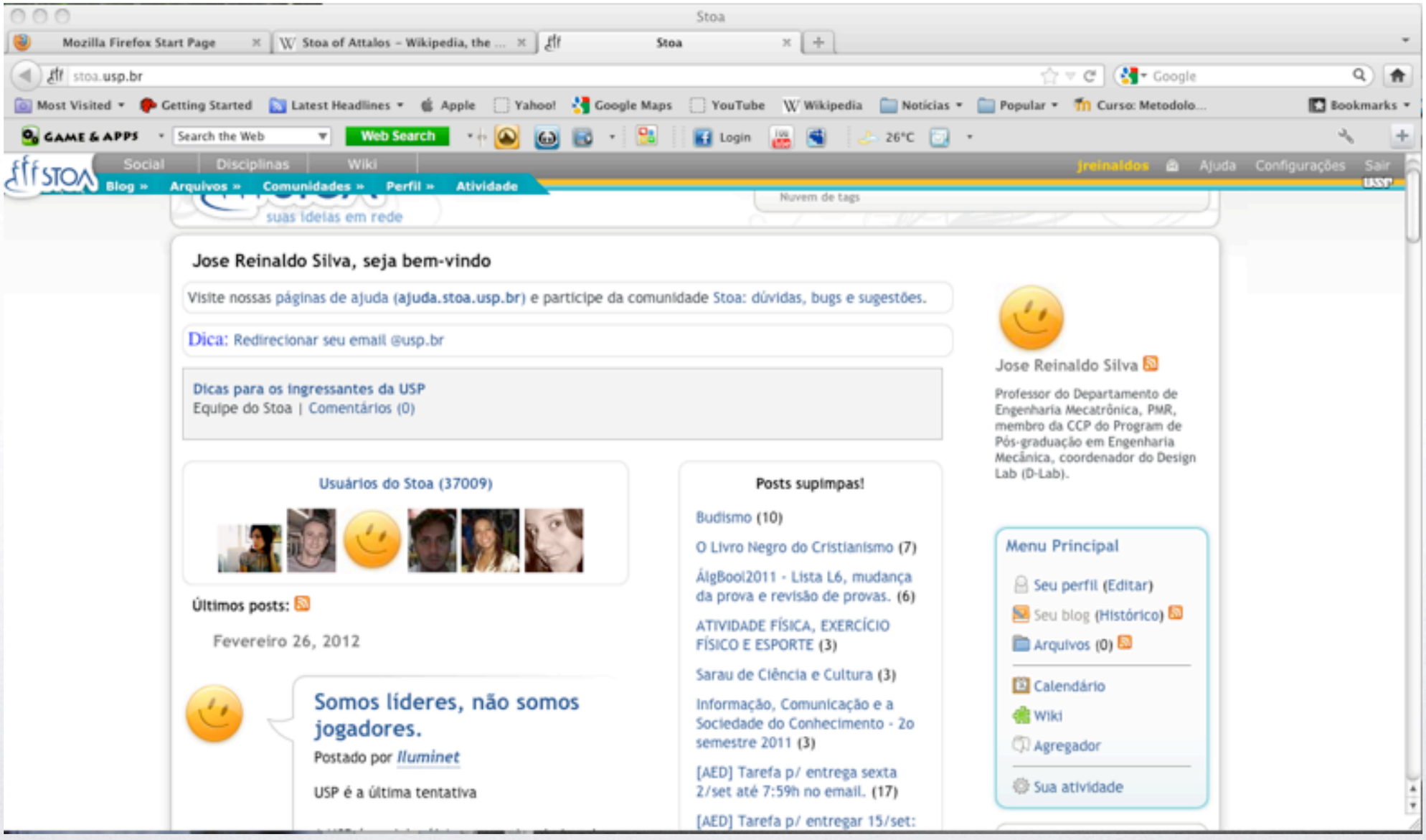
As Stoas eram espaços públicos onde as pessoas expunham quadros, obras de artesanato, vendiam produtos, víveres, declamavam, etc. Eram comuns na Grecia em vários períodos, incluindo o Helenístico.

O projeto Stoa-USP é uma stoa virtual onde a comunidade USP pode debater, postar opiniões, criar blogs, compartilhar noticias, grupos de discussão e também disciplinas, estas através do sistema Moodle de educação a distância. Vários cursos de graduação e pós-graduação se encontram no Stoa-USP icluindo PMR 5237.

Neste caso o sistema está acoplado diretamente com o sistema Janus e deve usar como e-mail o enderaço oficial no sistema USP. Os alunos ouvintes devem se registrar no sistema STOA e serão incluídos depois no sistema.

Stoa de Atallos, Grecia, 159-138 AC, reformado recentemente

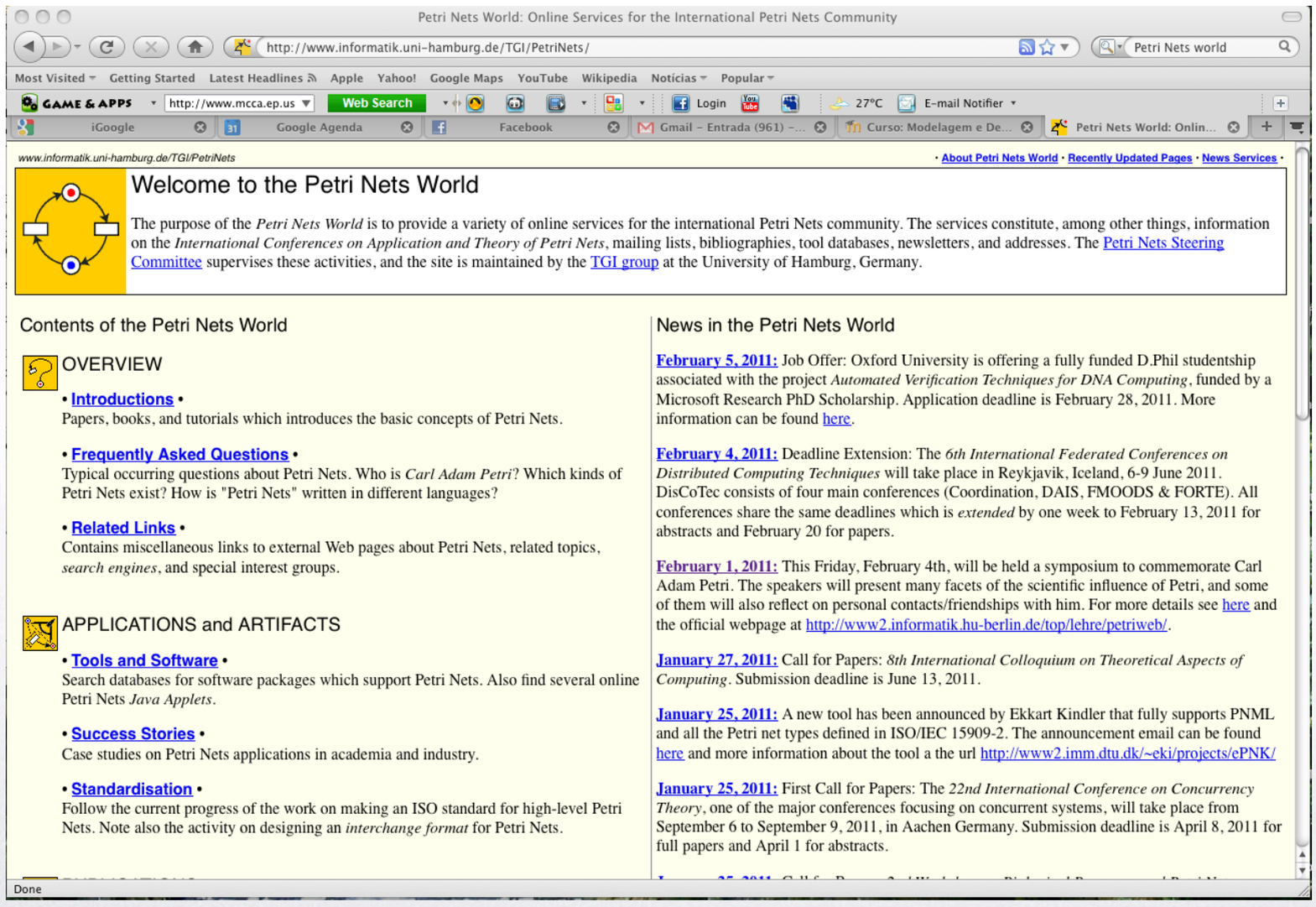
STOA - USP


Stoa
Mozilla Firefox Start Page
Stoa of Attalos – Wikipedia, the ...
Stoa
stoa.usp.br
Most Visited
Getting Started
Latest Headlines
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
Notícias
Popular
Bookmarks
Search the Web
Web Search
Login
26°C
Social
Disciplinas
Wiki
jreinaldos
Ajuda
Configurações
Sair
Blog »
Arquivos »
Comunidades »
Perfil »
Atividade
suas ideias em rede
Nuvem de tags
Jose Reinaldo Silva, seja bem-vindo
Visite nossas páginas de ajuda (ajuda.stoa.usp.br) e participe da comunidade Stoa: dúvidas, bugs e sugestões.
Dica: Redirecionar seu email @usp.br
Dicas para os ingressantes da USP
Equipe do Stoa | Comentários (0)
Usuários do Stoa (37009)
Últimos posts:
Fevereiro 26, 2012
Somos líderes, não somos jogadores.
Postado por Iluminet
USP é a última tentativa
Posts supimpas!
Budismo (10)
O Livro Negro do Cristianismo (7)
ÁlgBool2011 - Lista L6, mudança da prova e revisão de provas. (6)
ATIVIDADE FÍSICA, EXERCÍCIO FÍSICO E ESPORTE (3)
Sarau de Ciência e Cultura (3)
Informação, Comunicação e a Sociedade do Conhecimento - 2o semestre 2011 (3)
[AED] Tarefa p/ entrega sexta 2/set até 7:59h no email. (17)
[AED] Tarefa p/ entregar 15/set:
Jose Reinaldo Silva
Professor do Departamento de Engenharia Mecatrônica, PMR, membro da CCP do Program de Pós-graduação em Engenharia Mecânica, coordenador do Design Lab (D-Lab).
Menu Principal
Seu perfil (Editar)
Seu blog (Histórico)
Arquivos (0)
Calendário
Wiki
Agregador
Sua atividade

Suporte a disciplinas :
Moodle STOA

Referência na Internet:
Petri Nets World

http://www.informatik.uni-hamburg.de/TGI/PetriNets/



Ferramentas de software

PIPE (Windows, Linux, Mac)

HPSIM (Windows)

Ferramentas de software

PIPE (Windows, Linux, Mac)

~~HPSIM (Windows)~~

CPN Tools

Ferramentas de software

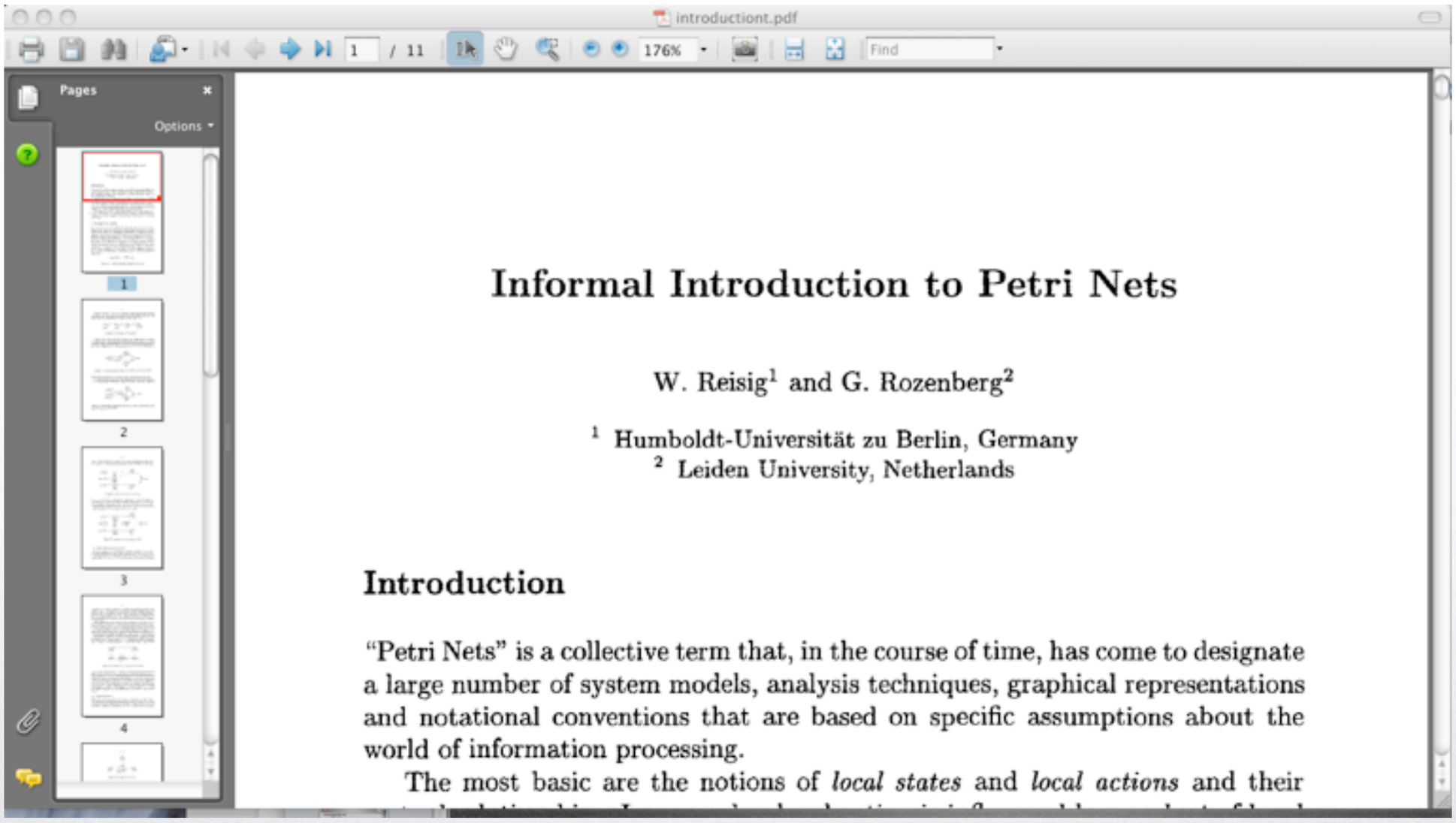
PIPE (Windows, Linux, Mac)

HPSIM (Windows)

CPN Tools

GHENeSys

Leitura da semana



Fim