PMR 5237
Modelagem e Design de Sistemas Discretos em Redes de Petri

Aula 13: Modelagem em Redes de Petri - Novas Perspectivas
Prof. José Reinaldo Silva
reinaldo@usp.br
Use of Petri Nets in Design

- Requirements
- Model Analysis
- Requirements Model
- Design Model
- Verification/Validation
- Design
**PN Basic Properties**

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).

**Fairness Invariants**
Going Further with Discrete Events Modeling

• Analysing a larger set of properties;
• Scaling the analysis process (hierarchies, etc.);
• Improve abstraction and theoretical level;
• Providing more sophisticated support environments.
Analysis in CPN Nets
Theoretical and practical reasons to use CPN Nets

- Improve abstraction (functional programming)
- Improve the formal semantics of PN
- Sound execution based sound support tools
Implementing Coloured Petri Nets
Using a Functional Programming Language

LARS MICHAEL KRISTENSEN*
SØREN CHRISTENSEN
Department of Computer Science, University of Aarhus, Aabogade 34, DK-8200 Aarhus N, Denmark

Abstract. Coloured Petri Nets (CPNs) are a graphically oriented modelling language for concurrent systems based on Petri Nets and the functional programming language Standard ML. Petri Nets provide the primitives for modelling concurrency and synchronisation. Standard ML provides the primitives for modelling data manipulation and for creating compact and parameterisable CPN models.

Functional programming and Standard ML have played a major role in the development of CPNs and the CPN computer tools supporting modelling, simulation, verification, and performance analysis of concurrent systems.

At the modelling language level, Standard ML has extended Petri Nets with the practical expressiveness required for modelling systems of the size and complexity found in typical industrial projects. At the implementation level, Standard ML has been used to implement the formal semantics of CPNs that provide the theoretical foundation of the CPN computer tools.

This paper provides an overview of how functional programming and Standard ML are applied in the CPN modelling language and the supporting computer tools. We give a detailed presentation of the key algorithms and techniques used for implementing the formal semantics of CPNs, and we survey a number of case studies where CPNs have been used for the design and analysis of systems. We also demonstrate how the use of a Standard ML programming environment has allowed Petri Nets to be used for the implementation of systems.

Keywords: distributed and concurrent computation, implementation techniques, programming environments and tools, Coloured Petri Nets, high level Petri Nets, Petri Nets

1. Introduction

An increasing number of system development projects are concerned with concurrent systems. Examples of these range from large scale systems in the areas of telecommunication and applications based on WWW technology, to medium or small scale systems, in the area of embedded systems. The development of such systems is complex, a main reason being that the execution of a concurrent system consisting of a number of independent but co-operating processes may proceed in many different ways, e.g., depending on messages lost, the speed of the processes involved, and the time at which input is received and processed.
What are the drawbacks?

CPN was designed to improve semantic analysis but not to fit together with classic PN.

Automation of the analysis of some structural and behavioural properties was lost.

Improve the design approach but detach from control and direct implementation.
Use of Petri Nets in Design

- Requirements
  - CPN
- Model Analysis
  - P/T
- Req. Analysis
- Requirements Model
- Design Model
- Verification/Validation
- Design
That implies in providing a better matching between classic and high level methods.
15909-1: Estabelece uma definição única de rede Place/Transition (P/T) e de rede de alto nível (HLPN) [ISO/IEC 2004]

15909-2: Define um formato de transferência para o intercâmbio de modelos em RdPs (PNML) [ISO/IEC 2011]

15909-3: Será dedicada à padronização das extensões das RdP, incluindo a hierarquia, o tempo e as funcionalidades estocásticas [Hillah et al. 2006]

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO’s adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 7, Software and systems engineering.

This second edition cancels and replaces the first edition (ISO/IEC 15909-1:2004), which has been technically revised.

The main change compared to the previous edition is as follows:

— a complete redrafting of the concepts and definitions of Petri nets and Petri net types in a simpler, modular and incremental way.

A list of all parts in the ISO/IEC 15909 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.
Basic concepts of HLPN

An HLPN net will be defined by...

- a net graph $NG$ that defines its syntax,
- an underlying algebraic specification that defines the semantic domain,
- a net inscription mapping syntactic entities to their semantic denotation
An HLPN is a structure:

\[ N = (NG, \text{Sig}, V, H, \text{Type}, \text{AN}, M_0). \]

- \( NG = (P, T; F) \) is a net graph, with:
  - \( P \) a finite set of nodes, called places;
  - \( T \) a finite set of nodes, called transitions, disjoint from \( P(P \cap T = \emptyset) \); and
  - \( F \subseteq (P \times T) \cup (T \times P) \) a set of directed edges called arcs, known as the flow relation.
- \( \text{Sig} = (S, O) \) is a Boolean signature, where \( S \) is a set of sorts and where \( O \) is a set of operators defined in the Annex A of ISO/IEC (2002).
- \( V \) is an \( S \)-indexed set of variables, disjoint from \( O \).
- \( H = (S_H, O_H) \) is a many-sorted algebra for the signature \( \text{Sig} \), defined in this list:
  - \( \text{Type} : P \to S_H \) is a function that assigns types to places.
  - \( \text{AN} = (A, TC) \) is a pair of net annotations.
    - \( A : F \to \text{TERM}(O \cup V) \) such that for all \((p, t), (t, p) \in F \) and all bindings \( \alpha, \text{Val}_\alpha(A(p, t)), \text{Val}_\alpha(A(t', p)) \in \mu \text{Type}(p) \). \( \text{TERM}(O \cup V), \alpha, \text{Val}_\alpha \) and \( \mu \text{Type}(p) \) are defined in Annex A of ISO/IEC (2002). \( A \) is a function that annotates each arc with a term that when evaluated (for some binding) results in a multiset over the associated place’s type.
    - \( TC : T \to \text{TERM}(O \cup V)_{\text{Bool}} \) is a function that annotates transitions with Boolean expressions.
  - \( M_0 : P \to \bigcup_{p \in P} \mu \text{Type}(p) \) such that \( \forall p \in P, M_0(p) \in \mu \text{Type}(p) \) is the initial marking function that associates a multiset of tokens (of the correct type) with each place.
Definition 4.3.1. A coloured Petri net (CPN) is defined by a tuple $\mathcal{N} = \langle P, T, \text{Pre}, \text{Post}, C, cd \rangle$, where

- $P$ is a finite set (the set of places of $\mathcal{N}$),
- $T$ is a finite set (the set of transitions of $\mathcal{N}$), disjoint from $P$,
- $C$ is the set of colour classes,
- $cd : P \cup T \to C$ is the colour domain mapping, and
- $\text{Pre}, \text{Post} \in \mathcal{B}^{\left|P\right| \times \left|T\right|}$ are matrices (the backward and forward incidence matrices of $\mathcal{N}$) such that $\text{Pre}[p, t] : cd(t) \to \text{Bag}(cd(p))$ and $\text{Post}[p, t] : cd(t) \to \text{Bag}(cd(p))$ are mappings for each pair $(p, t) \in P \times T$.

$\mathcal{B}$ can be taken as the set of mappings of the form $f : cd(t) \to \text{Bag}(cd(p))$. Again, $C = \text{Post} - \text{Pre}$ is called the incidence matrix.
**Definition:** A Coloured Petri Net is a tuple \( \text{CPN} = (\Sigma, P, T, A, N, C, G, E, I) \) satisfying the following requirements:

(i) \( \Sigma \) is a finite set of non-empty types, called **colour sets**.

(ii) \( P \) is a finite set of **places**.

(iii) \( T \) is a finite set of **transitions**.

(iv) \( A \) is a finite set of **arcs** such that:

- \( P \cap T = P \cap A = T \cap A = \emptyset \).

(v) \( N \) is a **node** function. It is defined from \( A \) into \( P \times T \cup T \times P \).

(vi) \( C \) is a **colour** function. It is defined from \( P \) into \( \Sigma \).
(vii) $G$ is a **guard** function. It is defined from $T$ into expressions such that:

- $\forall t \in T: \text{Type}(G(t)) = \text{Bool} \land \text{Type}(\text{Var}(G(t))) \subseteq \Sigma$.

(viii) $E$ is an **arc expression** function. It is defined from $A$ into expressions such that:

- $\forall a \in A: \text{Type}(E(a)) = C(p(a))_{MS} \land \text{Type}(\text{Var}(E(a))) \subseteq \Sigma$

where $p(a)$ is the place of $N(a)$.

(ix) $I$ is an **initialization** function. It is defined from $P$ into closed expressions such that:

- $\forall p \in P: \text{Type}(I(p)) = C(p)_{MS}$. 
A marking in a HLPN net is a multiset defined by

\[ M: P \rightarrow \bigcup_{p \in P} \mu \text{ Type}(p) \text{ such that } \forall p \in P, M(p) \in \mu \text{ Type}(p). \]
Definition 35

Seja uma rede P/T com estrutura \( N, PT = (N, K, W, M_0) \). e uma bijeção (equivalência) \( \rho \) que preserva o sort. Chama-se rede quociente em relação a \( \rho \) ao sistema \( \tilde{PT} = (\tilde{N}, \tilde{K}, \tilde{W}, \tilde{M}_0) \) que tem a mesma dinâmica que a rede original.
Basic High Level Net

- lugares possuem marcas individualizadas
- transições ocorrem em diferentes modos restritos à regra de localidade (agir somente sobre o seu pre-set e pós-set)

\[ p_1 = \{a_1, a_2\} \]
\[ p_2 = \{a_3, a_4\} \]
\[ t_1 = \{s_1, s_2\} \]
\[ t_2 = \{s_3, s_4, s_5, s_6\} \]

\[ \alpha_1 = \begin{cases} 
\text{if } x = a \\
\text{then } y = a \\
\text{else if } x = b \\
\text{then } y = b 
\end{cases} \]

\[ \alpha_2 = \begin{cases} 
\text{if } y = a \\
\text{then } (x = a) \lor (x = b) \\
\text{else if } y = b \\
\text{then } (x = a) \lor (x = b) 
\end{cases} \]

\[ \text{P} = \{a, b_2\} \]
Formalmente, cada rede P/T está relacionada com uma rede quociente...

**Definition 34**

Seja uma rede place/transition $N$ e seja o seu domínio $X = P \cup T$. Existe uma equivalência $\rho$ entre a rede $N$ e sua rede quociente $\tilde{N}$ tal que:

i) $\forall x \in X, \exists \bar{x}$ que denota uma classe de equivalência $\{y \in X | x \rho y\}$.

ii) Seja $Y \subseteq X$, então $\tilde{Y} := \{\bar{x} | x \in Y\}$.

iii) $\rho$ preserva o sort, isto é, $\rho \cap (P \times T) = \emptyset$.

iv) A relação de fluxo $\tilde{F}$ sobre o domínio $\tilde{X}$ é definida por,

$$\bar{x} F \bar{y} \iff \exists x \bar{l} \in \bar{x} \land \exists y \bar{l} \in \bar{y} | x F y$$

v) Denota-se a nova rede $(\tilde{P}, \tilde{T}; \tilde{F})$ de $\tilde{N}$. 
Veremos mais adiante um processo chamado unfolding ("desdobramento"), que consiste justamente em desfazer os dobramentos resultantes do processo de fatoração.

Trata-se portanto em desfazer o colapso das simetrias que levaram aos dobramentos, e à identificação das marcas nas redes de alto nível/coloridas. Portanto, no unfolding cada tipo de marca deve ter sua própria sub-rede, simétrica e funcionalmente equivalentes entre si. A rede resultante deste processo é novamente uma rede P/T.

Dada uma rede P/T é sempre possível dobrá-la e obter com isso a correspondente rede quociente. Por sua vez a rede quociente pode ser trabalhada (baseada em sorts e teoria de tipos) para obter a rede de alto nível equivalente. É possível também traçar o caminho inverso e, à partir da rede de alto nível, obter a rede equivalente P/T, usando para isso as técnicas de unfolding já descritas.
Definition 4.11. Let \( \overline{PT} = (\tilde{N}, \tilde{K}, \tilde{W}, \tilde{M}_{\text{in}}) \) be a quotient system. The basic high level net \( (\tilde{N}, D, \Phi, \tilde{W}, \tilde{K}, \tilde{M}_{\text{in}}) \), where \( D \) and \( \Phi \) are defined according to (4.10), is called the BHL associated with \( \overline{PT} \).
**Definition 4.13.** The *unfolded system associated* with a basic high level net BHL is the PT-net $\overline{\text{BHL}} = (\tilde{P}, \tilde{T}, \tilde{F}, \tilde{W}, \tilde{K}, \tilde{M}_{\text{in}})$ as defined above.

It is immediately verified that a strict BHL gives rise to an *elementary net system.*

(i) Let $\tilde{\mathbf{P}}_T$ be a quotient of a PT-net $\mathbf{P} T$. Let $\tilde{\mathbf{P}}_T$ be the unfolding of $\tilde{\mathbf{P}}_T$ according to Definition 4.13. Then, except for names of elements, $\mathbf{P} T$ and $\tilde{\mathbf{P}}_T$ are identical. In particular, transition-enabling and -occurrence in both systems coincide.

(ii) Let $\mathbf{BHL}$ be the PT-net obtained from a basic high-level net $\mathbf{BHL}$ according to Definition 4.13. Let $\mathbf{BHL}$ be the quotient of $\mathbf{BHL}$ with respect to the equivalence $\rho$ given by

$$(x, y) \rho (x', y') : \iff x = x'.$$

Then the systems $\mathbf{BHL}$ and $\mathbf{BHL}$ are identical, except for names of elements. In particular, transition-enabling and -occurrence coincide.
Use of Petri Nets in Design

- **P/T Requirements**
- **CPN**

- **CPN Model**
- **Analysis P/T**

- **Verification/Validation**

- **Req. Analysis**

- **P/T Requirements Model CPN**

- **Design**

- **P/T Design Model CPN**
Principles of High-Level Net Theory

Einar Smith

GMD, Sankt Augustin, Germany, esmith@gmd.de

Abstract. The paper gives an introduction to fundamentals and recent trends in the theory of high-level nets. High-level nets are first formally derived from low-level nets by means of a quotient construction. Based on a linear-algebraic representations, we develop an invariant calculus that essentially corresponds to the algebraic core of the well-known coloured nets. We demonstrate that the modelling power of high-level nets stems from the use of expressive symbolic annotation languages, where as a typical model we consider predicate-transition nets, both concrete models and net-schemes. As examples of specific high-level analysis-tools we discuss symbolic place-invariants and reachability-trees.
15909-1: Estabelece uma **definição única** de rede Place/Transition (P/T) e de rede de alto nível (HLPN) [ISO/IEC 2004]

**Redes Simétricas**

15909-2: Define um **formato de transferência** para o intercâmbio de modelos em RdPs (PNML) [ISO/IEC 2011]

15909-3: Será dedicada à padronização das **extensões** das RdP, incluindo a hierarquia, o tempo e as funcionalidades estocásticas [Hillah et al. 2006]

**Redes Orientadas a Objetos**
Object Petri Nets
Using the Nets-within-Nets Paradigm

Rüdiger Valk

Universität Hamburg, Vogt-Kölhn-Str.30, D-22527 Hamburg, Germany
valk@informatik.uni-hamburg.de

Abstract. The nets-within-nets paradigm provides an innovative modelling technique by giving tokens themselves the structure of a Petri net. These nets, called token nets or object nets, also support the object oriented modelling technique as they may represent real world objects with a proper dynamical behaviour. Between object nets and the surrounding net, called system net, various interaction mechanisms exist as well as between different object nets. This introduction into the field of object Petri nets starts with small examples and proceeds by giving formal semantics. Some of the examples are modelled within the formalism of the Renew tool. Finally the differences between reference and two kinds of value semantics are discussed.

1 Nets within Nets
Para além da estruturação

Redes **clássicas**
Hierárquicas e de alto Nível

→

Redes Orientadas a Objetos
Vantagens de uma abordagem orientada a objetos

• um modelo simples de concorrência e sincronização.
• semântica operacional que suporte simulação e execução.
• técnicas de análises.
• abstração
• refinamento
• encapsulamento
• reutilização e compartilhamento.
Redes de Petri e objetos

Propostas

- Tratamento por objetos do processo de geração da rede

- Exploração da dualidade e da hierarquia (e eventualmente da estruturação) ➔ herança simples

- Manutenção ou redefinição das propriedades das redes
O dilema Petri Nets versus objetos

Petri Nets inside object

X

Objects inside Petri Nets
Se considermos agora as redes de alto nível, estas podem ser representadas por redes por objetos onde a marca é também um objeto ("tipado"), significando um objeto onde somente a parte estática está definida. Neste caso reproduzimos as redes de alto nível ou coloridas.

Entretanto, se incluímos no objeto marca um comportamento distinto temos agora o que Rudiger Valk chama de “nets inside nets”, ou uma representação de segunda ordem.
Let X be a generic “agent”,

a) agent X at location A

b) mobile computer X in security environment A

agent X at location B

mobile computer X in security environment B

Fig. 1. Moving objects I

Fusão Petri Nets & Objetos

- A classe mãe não pode existir sem sacrificar a dualidade das redes, que gera a maioria das propriedades usadas em análise

- É difícil achar um significado para o polimorfismo nas redes

- Não é possível (ou pelo menos é difícil) achar um uso para a herança múltipla
<table>
<thead>
<tr>
<th>No.</th>
<th>Attribute</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>paradigm</td>
<td>state machine, algebra, process algebra, trace</td>
</tr>
<tr>
<td>2</td>
<td>formality</td>
<td>informal, semi-formal, formal</td>
</tr>
<tr>
<td>3</td>
<td>graphical representation</td>
<td>yes, no</td>
</tr>
<tr>
<td>4</td>
<td>object-oriented</td>
<td>yes, no</td>
</tr>
<tr>
<td>5</td>
<td>concurrency</td>
<td>yes, no</td>
</tr>
<tr>
<td>6</td>
<td>executability</td>
<td>yes, no</td>
</tr>
<tr>
<td>7</td>
<td>usage of variables</td>
<td>yes, no</td>
</tr>
<tr>
<td>8</td>
<td>non-determinism</td>
<td>yes, no</td>
</tr>
<tr>
<td>9</td>
<td>logic</td>
<td>yes, no</td>
</tr>
<tr>
<td>10</td>
<td>provability</td>
<td>yes, no</td>
</tr>
<tr>
<td>11</td>
<td>model checking</td>
<td>yes, no</td>
</tr>
<tr>
<td>12</td>
<td>event inhibition</td>
<td>yes, no</td>
</tr>
<tr>
<td>method name</td>
<td>paradigm</td>
<td>formality</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Action Systems</td>
<td>state transition</td>
<td>formal</td>
</tr>
<tr>
<td>B</td>
<td>state transition</td>
<td>formal</td>
</tr>
<tr>
<td>CASL</td>
<td>algebra</td>
<td>formal</td>
</tr>
<tr>
<td>Cleanroom &amp; JSD</td>
<td>traces &amp; process algebra</td>
<td>formal</td>
</tr>
<tr>
<td>COQ</td>
<td>state transition</td>
<td>formal</td>
</tr>
<tr>
<td>Estelle</td>
<td>state transition</td>
<td>formal</td>
</tr>
<tr>
<td>LOTOS</td>
<td>process algebra</td>
<td>formal</td>
</tr>
<tr>
<td>OMT &amp; B</td>
<td>state transition</td>
<td>formal</td>
</tr>
<tr>
<td>Petri Nets</td>
<td>state transition</td>
<td>formal</td>
</tr>
<tr>
<td>Petri Nets with Objects</td>
<td>state transition</td>
<td>formal</td>
</tr>
<tr>
<td>SART</td>
<td>state transition</td>
<td>informal &amp; semi-formal</td>
</tr>
<tr>
<td>SAZ</td>
<td>state transition</td>
<td>semi-formal &amp; formal</td>
</tr>
<tr>
<td>SCCS</td>
<td>process algebra</td>
<td>formal</td>
</tr>
<tr>
<td>SDL</td>
<td>state transition</td>
<td>formal</td>
</tr>
<tr>
<td>UML</td>
<td>state transition</td>
<td>informal &amp; semi-formal</td>
</tr>
<tr>
<td>VHDL</td>
<td>state transition</td>
<td>formal</td>
</tr>
<tr>
<td>Z</td>
<td>state transition</td>
<td>formal</td>
</tr>
<tr>
<td>method name</td>
<td>concurrency</td>
<td>executability</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Action Systems</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>B</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>CASL</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Cleanroom &amp; JSD</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>COQ</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Estelle</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>LOTOS</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>OMT &amp; B</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Petri Nets</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Petri Nets with Objects</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SART</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>SAZ</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>SCCS</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>SDL</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>UML</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>VHDL</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Z</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>
Fim