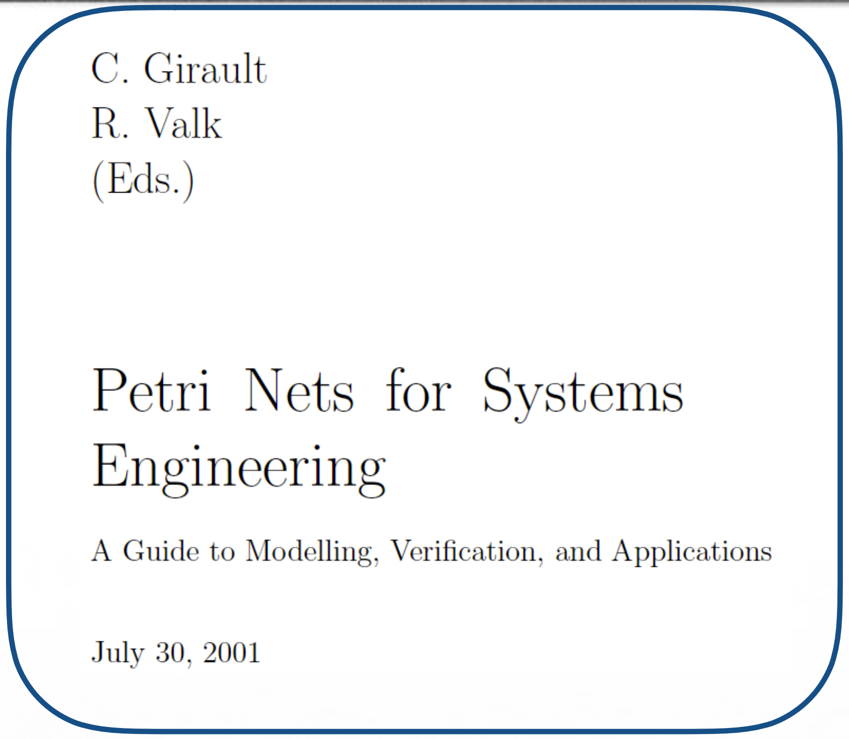
PMR 5237
Modelagem e Design de Sistemas Discretos em Redes de Petri

Aula 9 : Modelagem e análise de propriedades em RdP

Prof. José Reinaldo Silva
reinaldo@usp.br

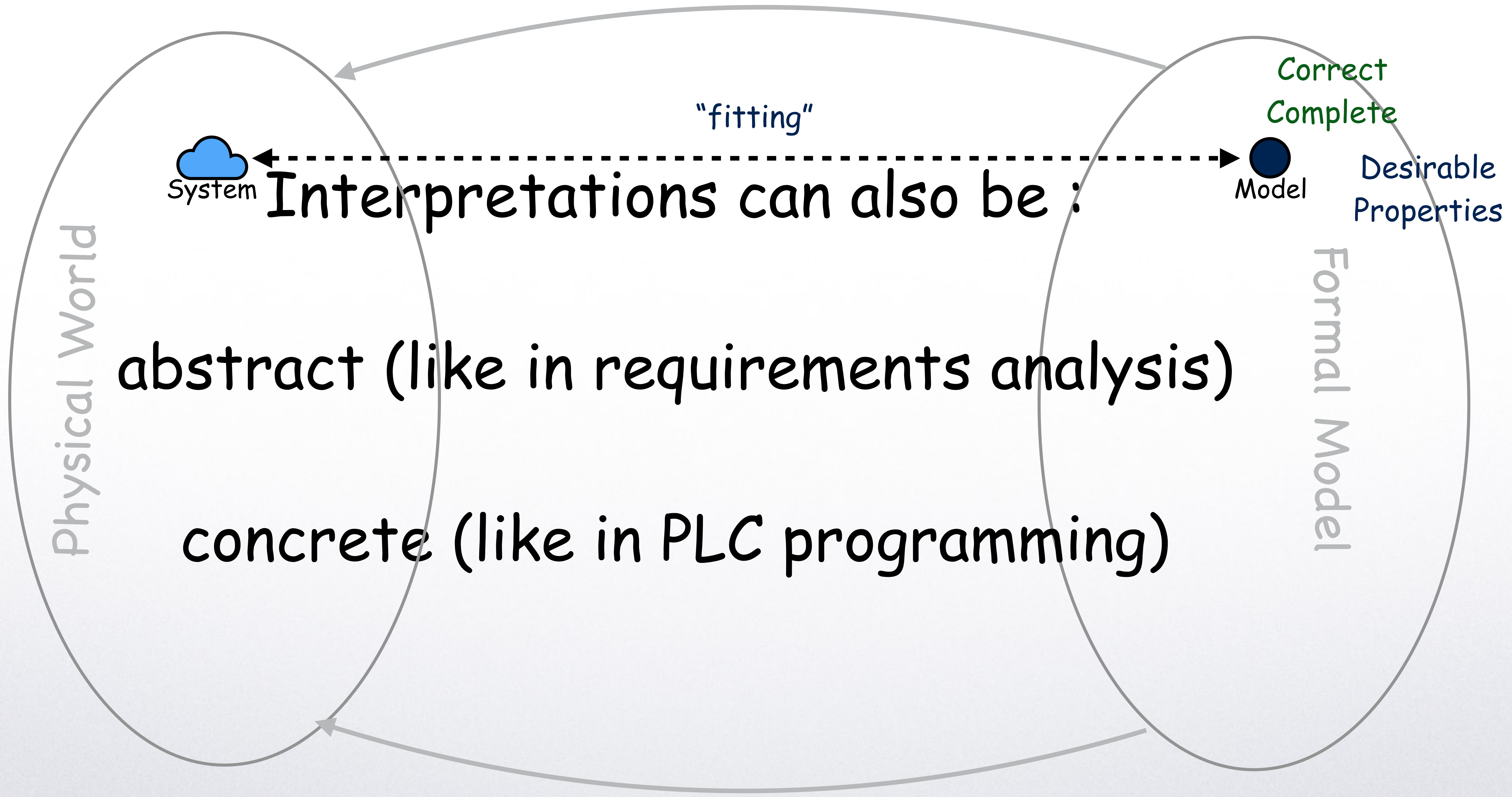

C. Girault
R. Valk
(Eds.)

Petri Nets for Systems Engineering

A Guide to Modelling, Verification, and Applications

July 30, 2001

The construction of Petri net models from informal requirement specifications is not a trivial task, and requires a great deal of modelling experience, as well as **knowledge of the techniques** aiding in model construction. As a result, a Petri net model may differ considerably from its original specification. This is especially true when large Petri net models of complex systems are involved.

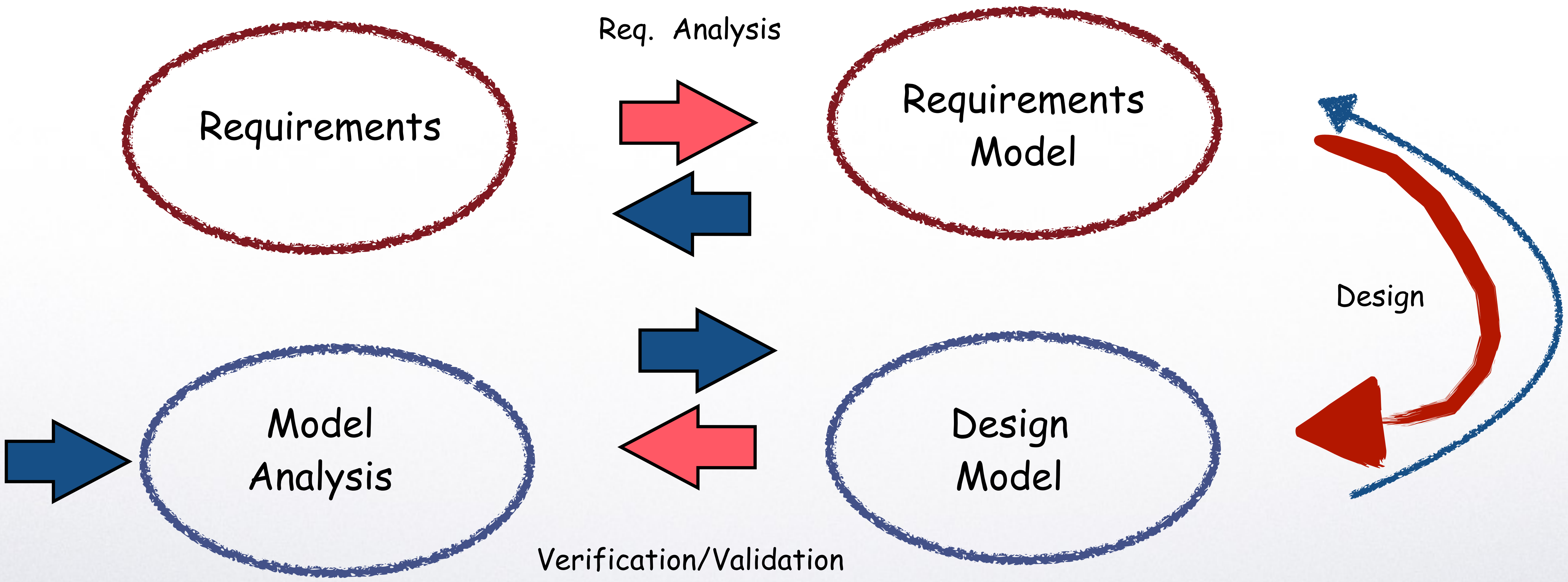


Interpretations can also be :

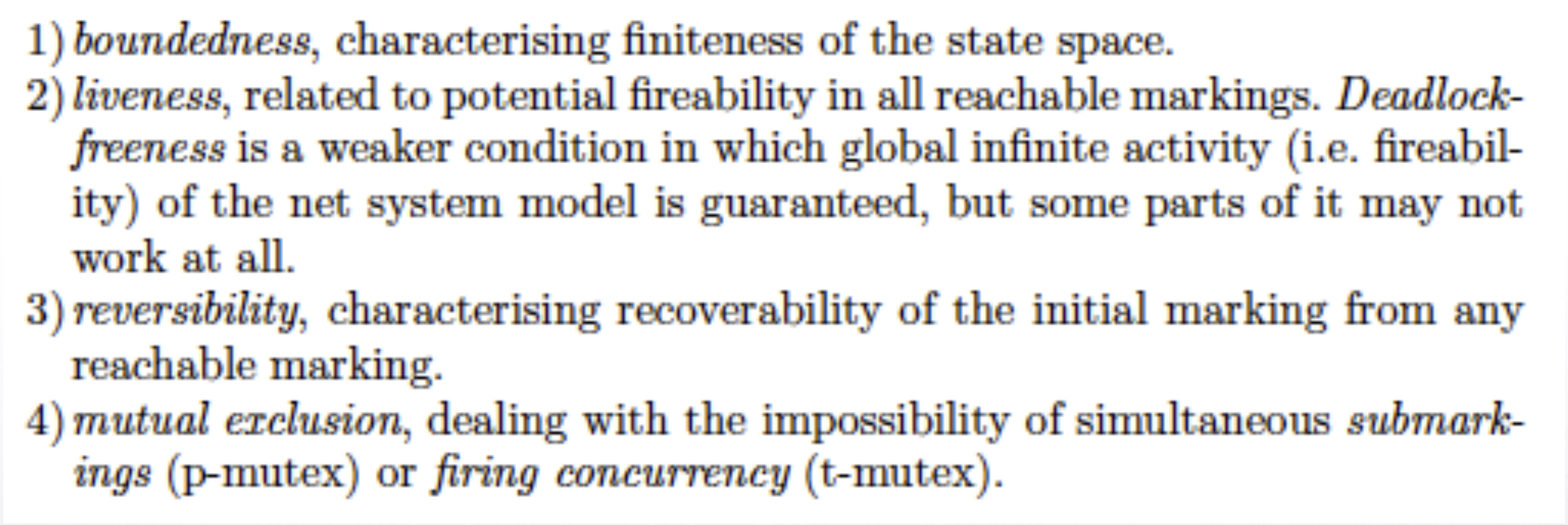
abstract (like in requirements analysis)

concrete (like in PLC programming)

Use of Petri Nets in Design



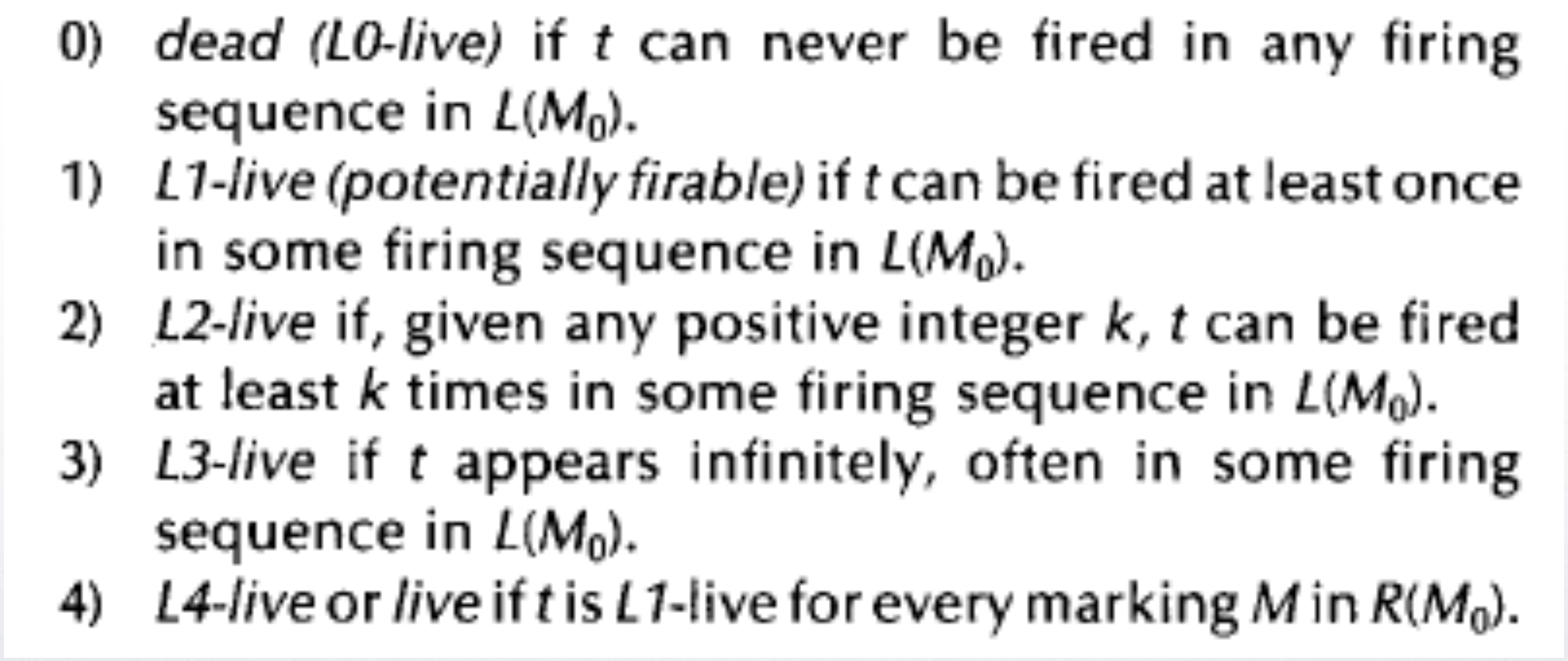
PN Basic Properties



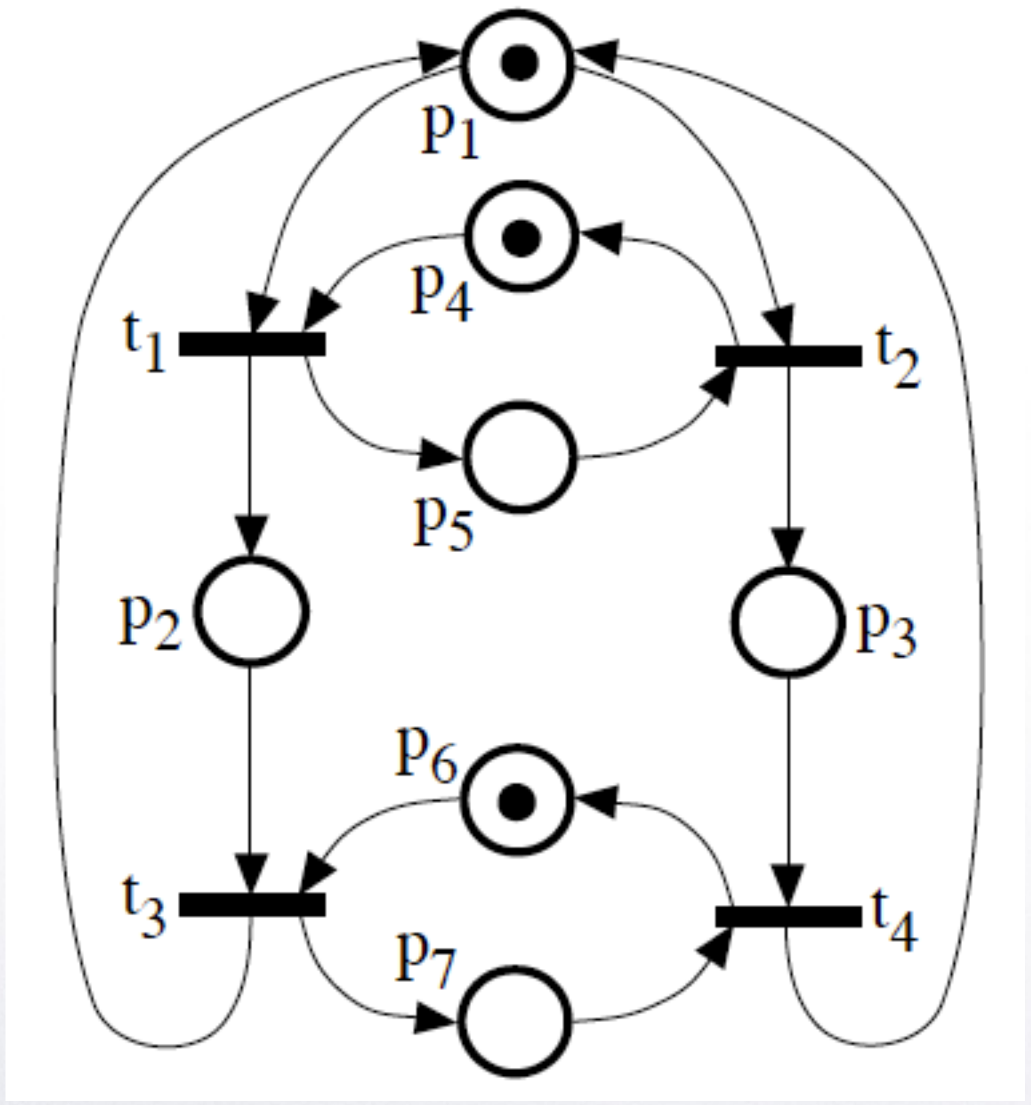
1) *boundedness*, characterising finiteness of the state space.
2) *liveness*, related to potential fireability in all reachable markings. *Deadlock-freeness* is a weaker condition in which global infinite activity (i.e. fireability) of the net system model is guaranteed, but some parts of it may not 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).

Lifeness (T. Murata)

A transition t is said to be...

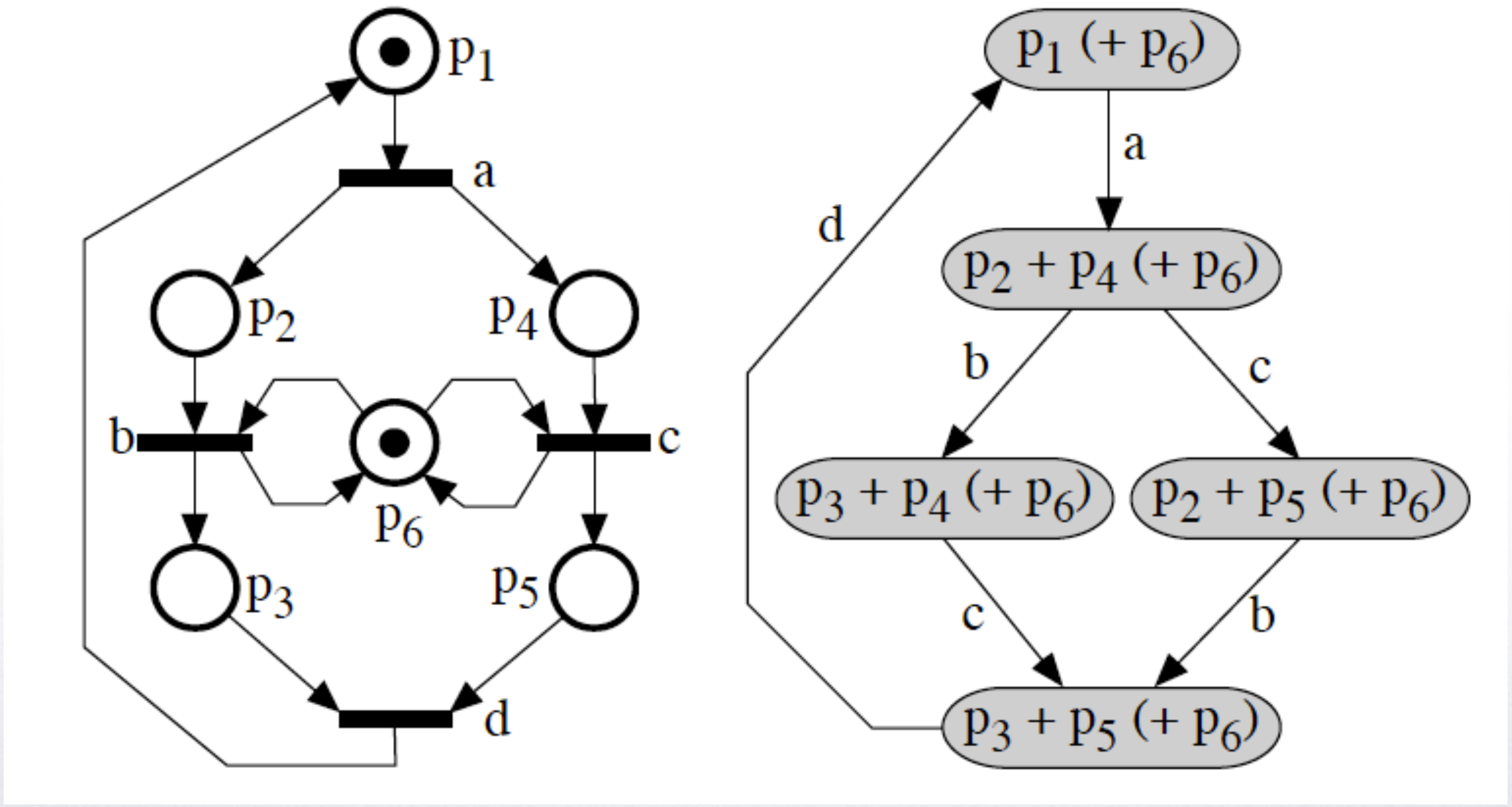


0) *dead (L0-live)* if *t* can never be fired in any firing sequence in $L(M_0)$.
1) *L1-live (potentially firable)* if *t* can be fired at least once in some firing sequence in $L(M_0)$.
2) *L2-live* if, given any positive integer *k*, *t* can be fired at least *k* times in some firing sequence in $L(M_0)$.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

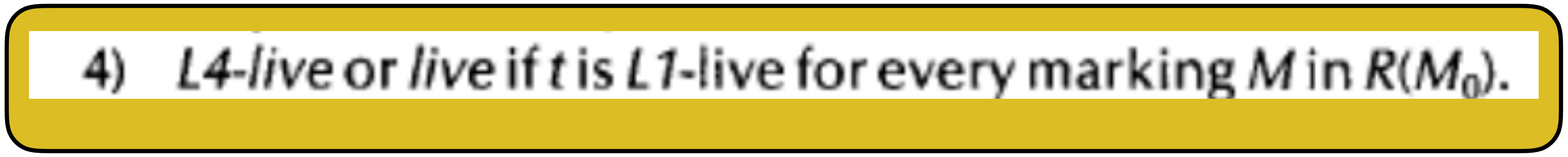


a live net system that by increasing the initial marking (e.g. m0[p5] = 1) can reach a deadlock state!

Reachability Analysis

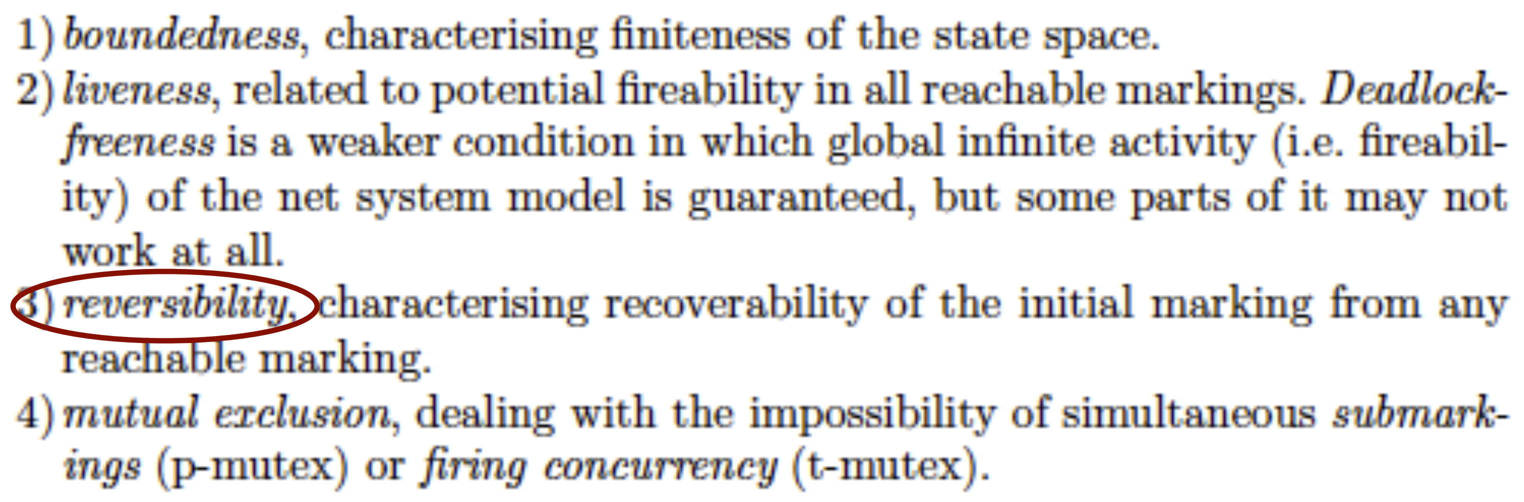


A very important goal for analysis is if a given system is deadlock-free.


4) *L4-live* or *live* if *t* is *L1*-live for every marking *M* in $R(M_0)$.

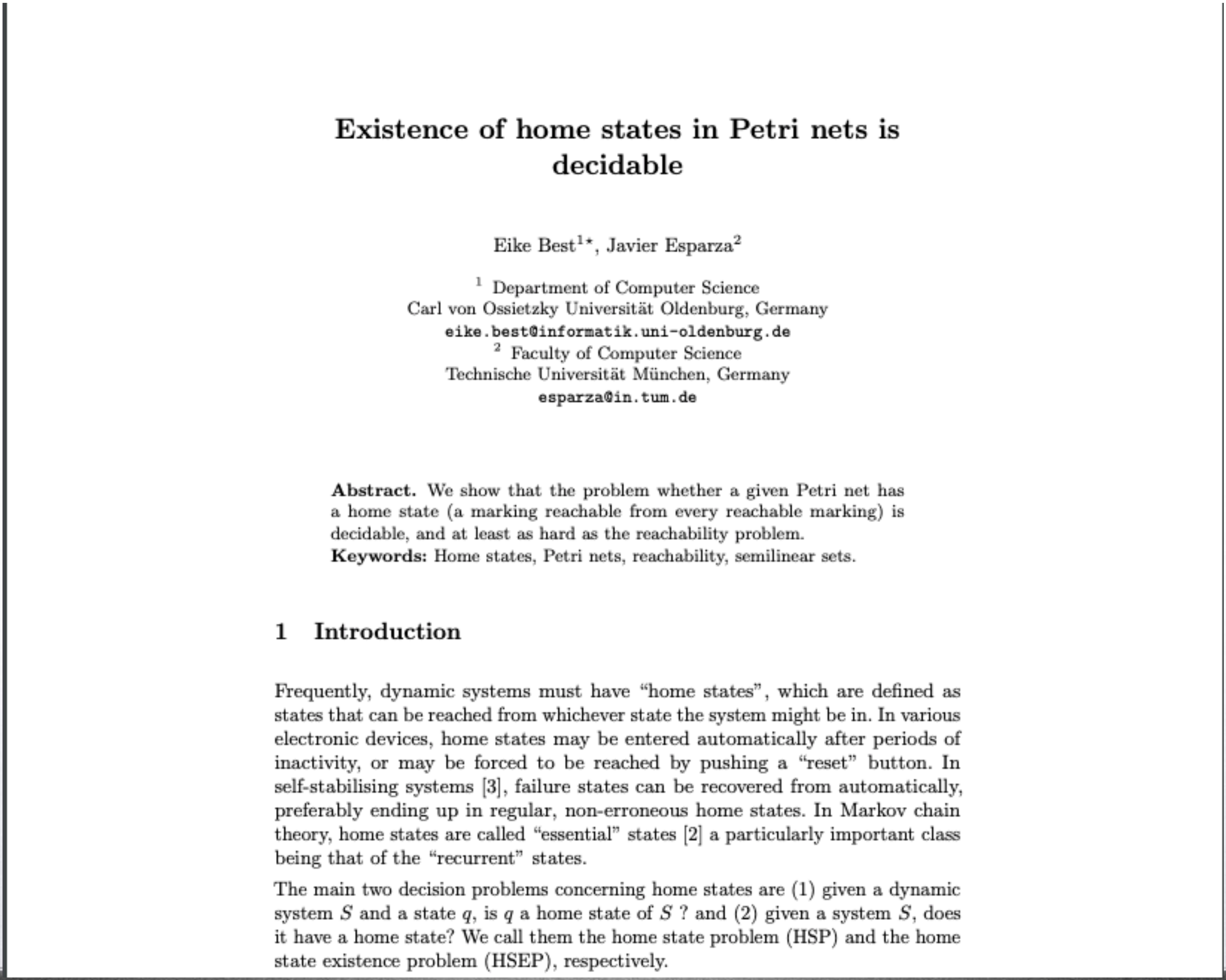
Def] A transition t is potentially fireable at a given marking m if there exists a transition iff the is a firing sequence σ leading to a marking m' in which t is enabled (i.e. $m \xrightarrow{\sigma} m' \geq Pre[P, t]$).

A P/T net is deadlock-free iff all its transitions are potentially fireable.



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).

Def] Let be the P/T net $N = (S, T, F; W, M_0)$. A generic state defined over this net X $\subset S$ is said to be a **home state** iff X can be reached from any reachable marking.



Existence of home states in Petri nets is decidable

Eike Best[1]*, Javier Esparza[2]

[1] Department of Computer Science
Carl von Ossietzky Universität Oldenburg, Germany
eike.best@informatik.uni-oldenburg.de
[2] Faculty of Computer Science
Technische Universität München, Germany
esparza@in.tum.de

Abstract. We show that the problem whether a given Petri net has a home state (a marking reachable from every reachable marking) is decidable, and at least as hard as the reachability problem.
Keywords: Home states, Petri nets, reachability, semilinear sets.

1 Introduction

Frequently, dynamic systems must have "home states", which are defined as states that can be reached from whichever state the system might be in. In various electronic devices, home states may be entered automatically after periods of inactivity, or may be forced to be reached by pushing a "reset" button. In self-stabilising systems [3], failure states can be recovered from automatically, preferably ending up in regular, non-erroneous home states. In Markov chain theory, home states are called "essential" states [2] a particularly important class being that of the "recurrent" states.

The main two decision problems concerning home states are (1) given a dynamic system S and a state q, is q a home state of S ? and (2) given a system S, does it have a home state? We call them the home state problem (HSP) and the home state existence problem (HSEP), respectively.

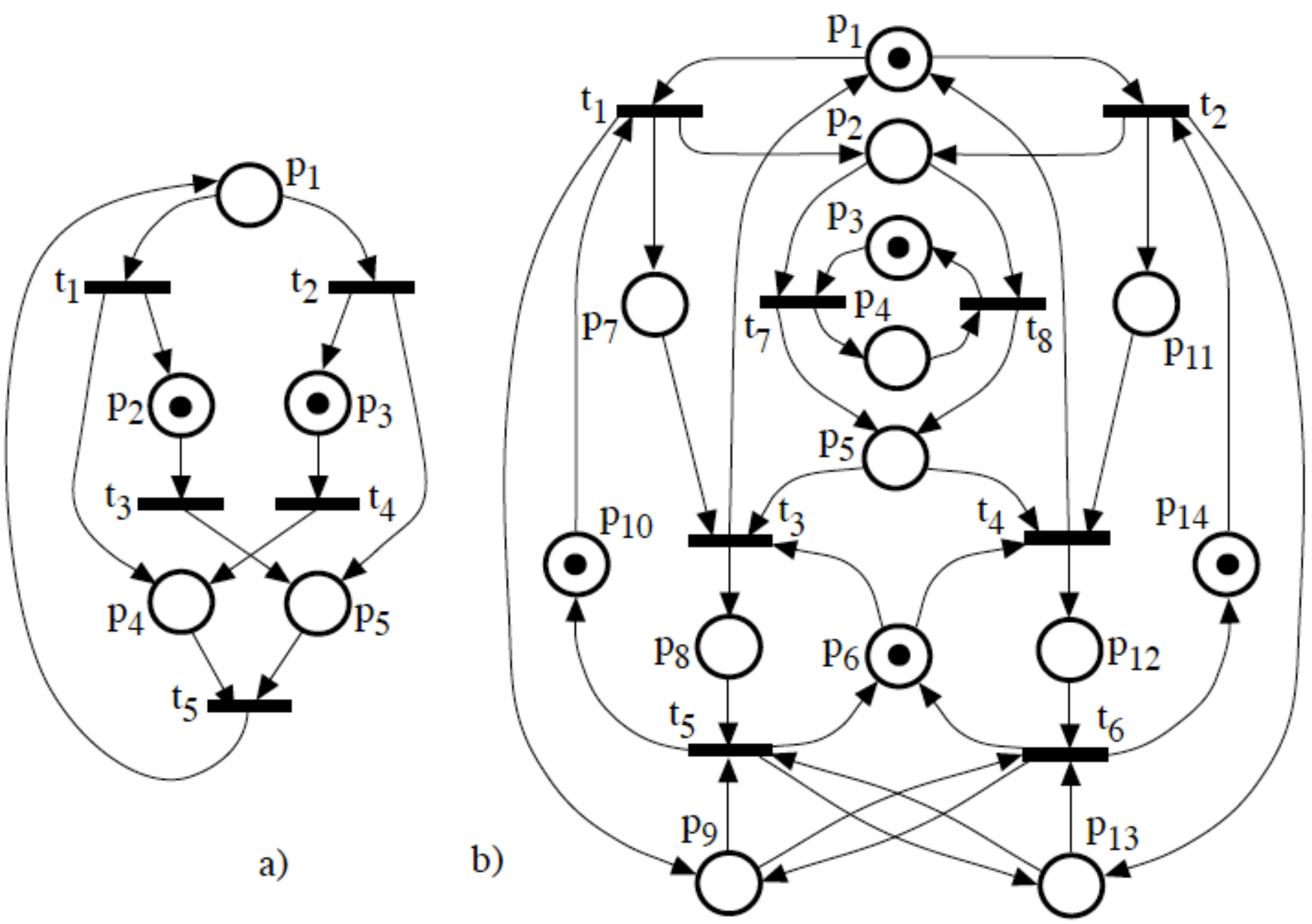
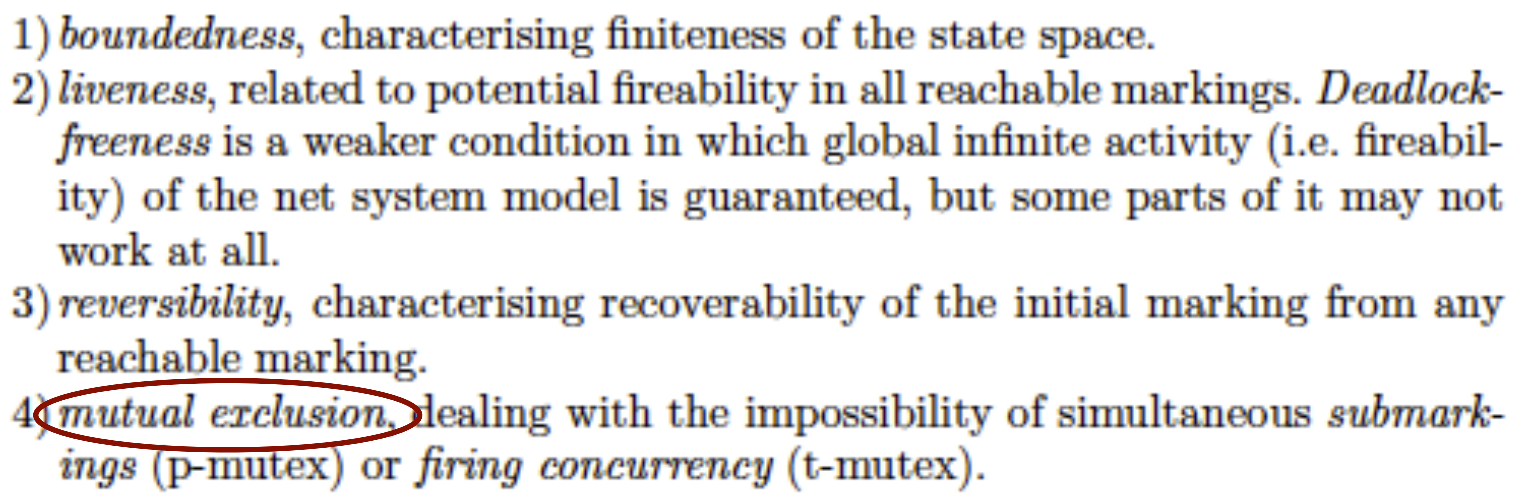


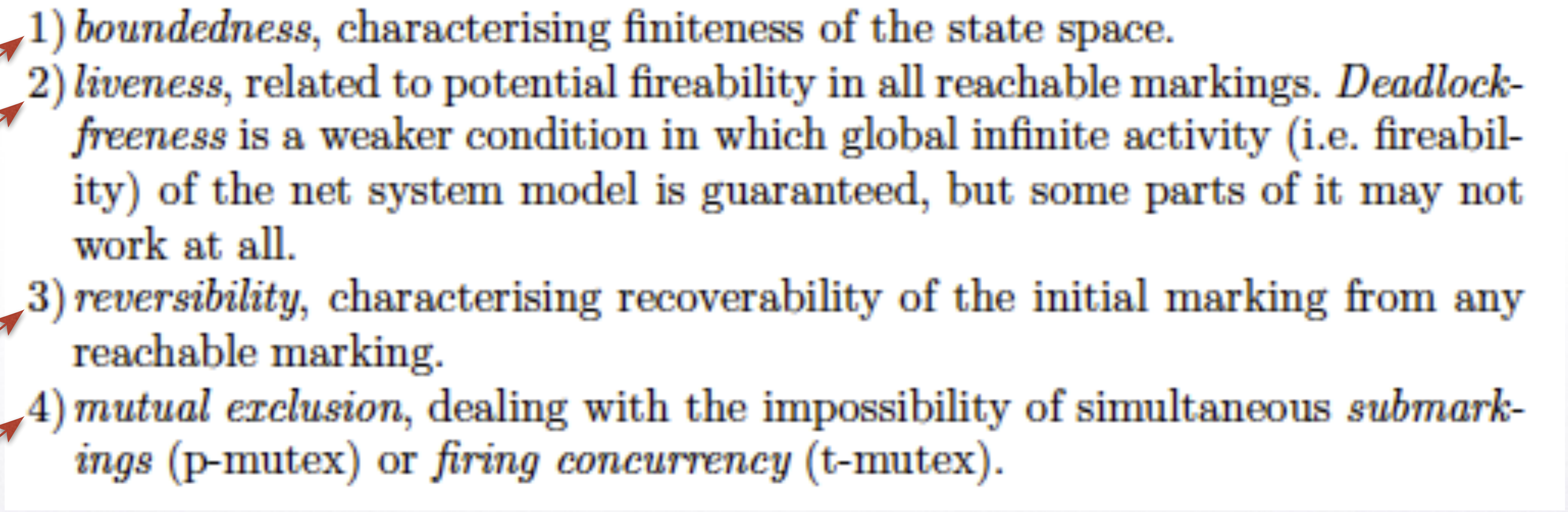
Fig. 5.3. On home states: (a) The initial marking is not a home state, but all successor markings are home states; (b) Net system that presents two livelocks, so there are no home states.



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).

In the beginning of our course it was proposed as an exercise to provide algorithms to analyse incidence matrix of classic net (therefore based on Linear Algebra) to detect arrangements where pre-set or pos-set of a transition (step) has non empty intersection.

PN Basic Properties

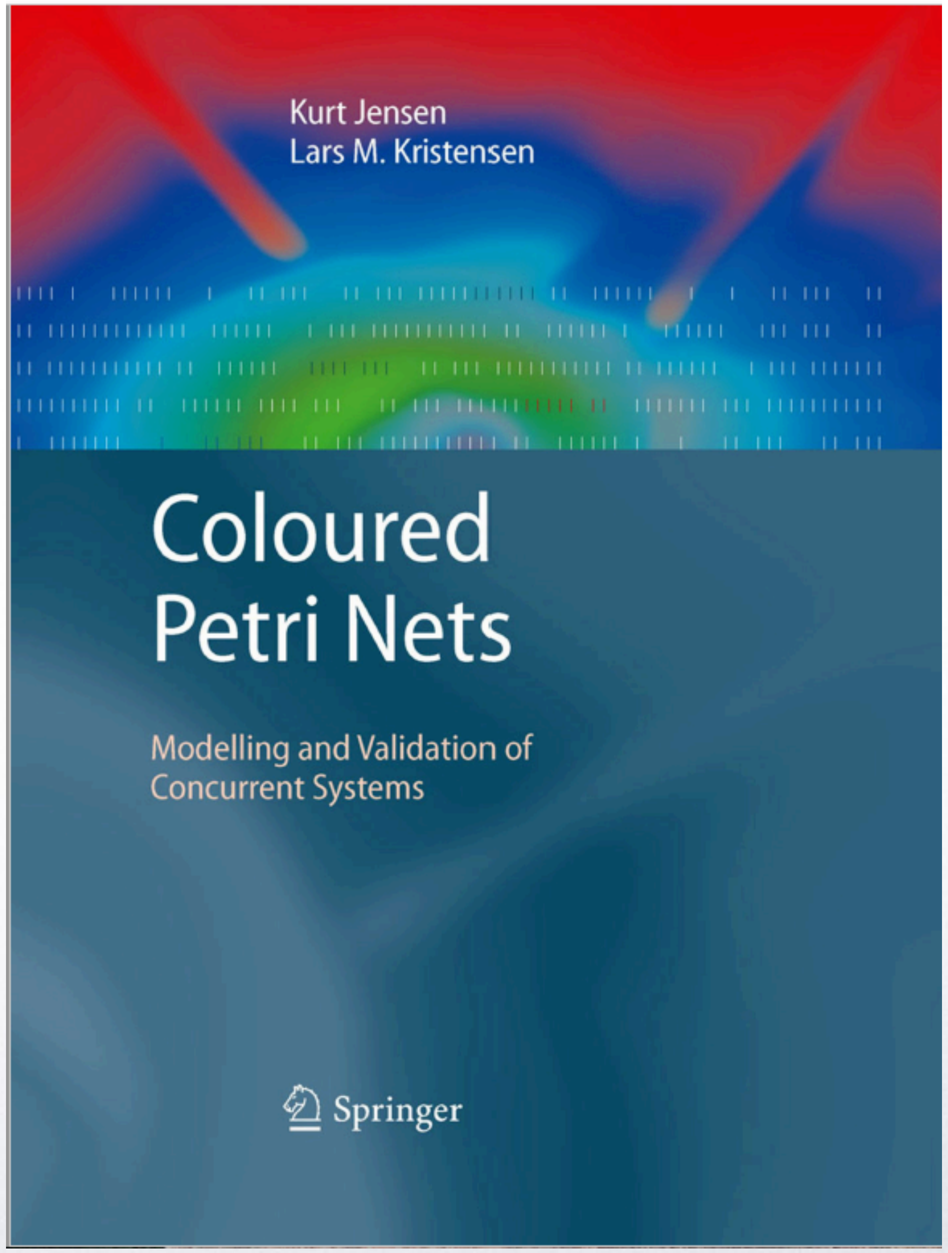


1) *boundedness*, characterising finiteness of the state space.
2) *liveness*, related to potential fireability in all reachable markings. *Deadlock-freeness* is a weaker condition in which global infinite activity (i.e. fireability) of the net system model is guaranteed, but some parts of it may not 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).

We should now investigate how to do this basic analyses in coloured/ high-level nets

CPN Property Analysis

ACM Computing Classification (1998): F.1, I.6, D.2.2, D.2.4
c Springer-Verlag Berlin Heidelberg 2009

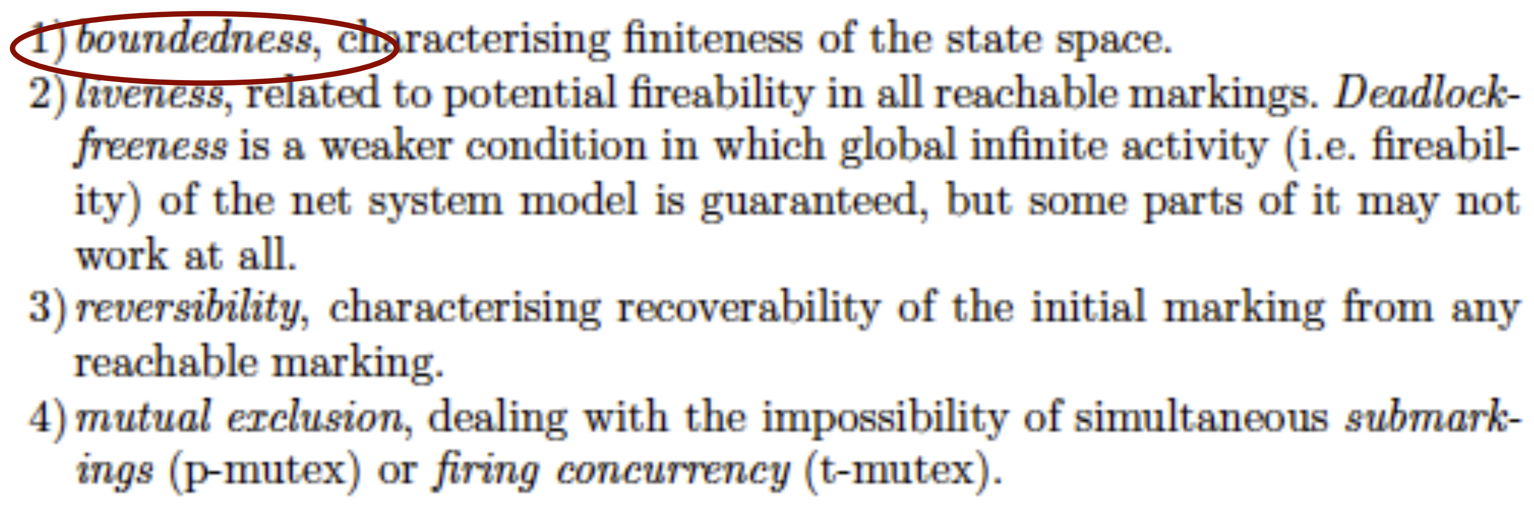

Kurt Jensen
Lars M. Kristensen

Coloured Petri Nets

Modelling and Validation of Concurrent Systems

Springer

Boudedness in CPN

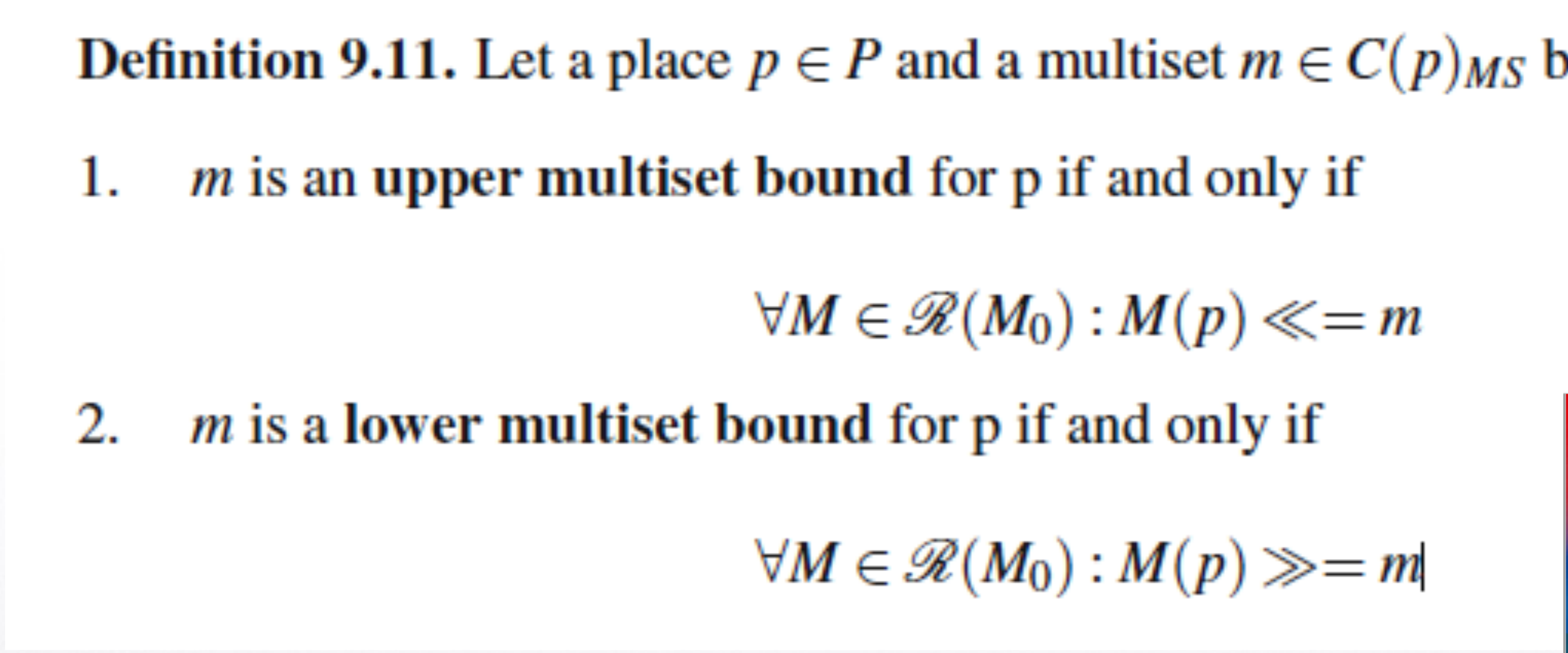


1) *boundedness*, characterising finiteness of the state space.
2) *liveness*, related to potential fireability in all reachable markings. *Deadlock-freeness* is a weaker condition in which global infinite activity (i.e. fireability) of the net system model is guaranteed, but some parts of it may not 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).

Let be CPN net N and a generic place p in this net and $|M(p)|$ the size of the multiset for making M. The place p is said bounded iff there is an integer n which is un upper bound of p, that is,

$$\forall M \epsilon \mathfrak{R}(M_0)\,.\,|M(p)| \leq n.$$

A CPN net is said to be bounded if all its place are bounded.



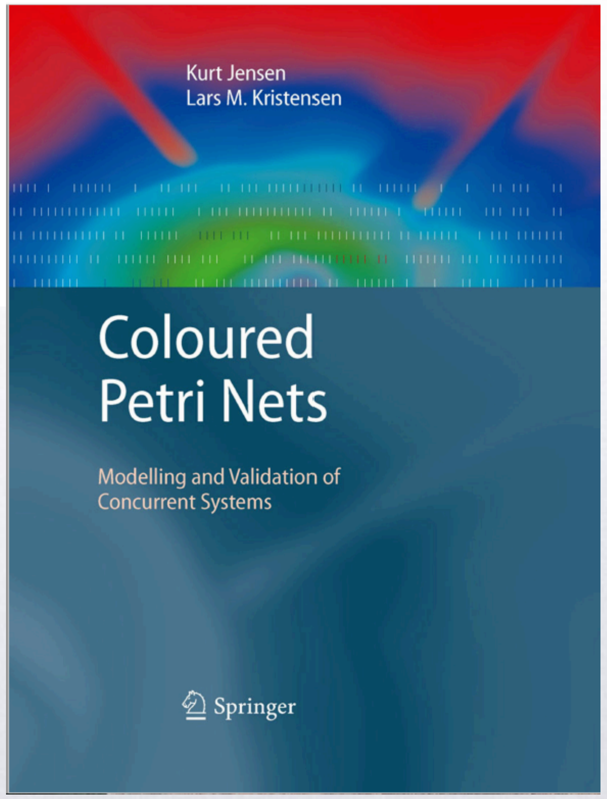
Definition 9.11. Let a place $p \in P$ and a multiset $m \in C(p)_{MS}$ be given.

1. m is an **upper multiset bound** for p if and only if

$$\forall M \in \mathscr{R}(M_0) : M(p) \ll= m$$

2. m is a **lower multiset bound** for p if and only if

$$\forall M \in \mathscr{R}(M_0) : M(p) \gg= m$$



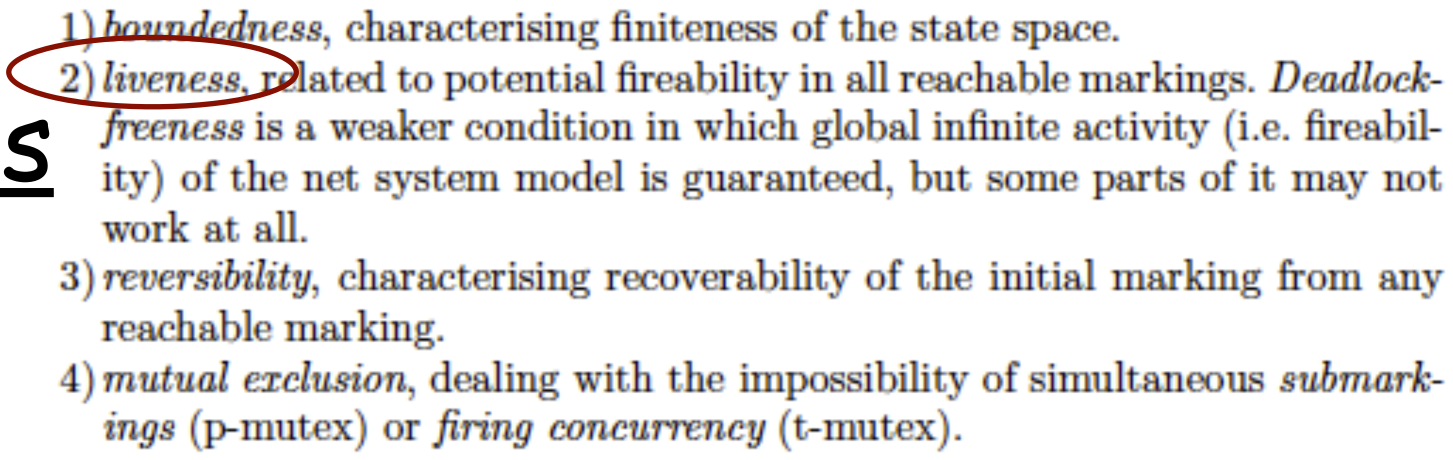
Kurt Jensen
Lars M. Kristensen

Coloured
Petri Nets

Modelling and Validation of
Concurrent Systems

Springer

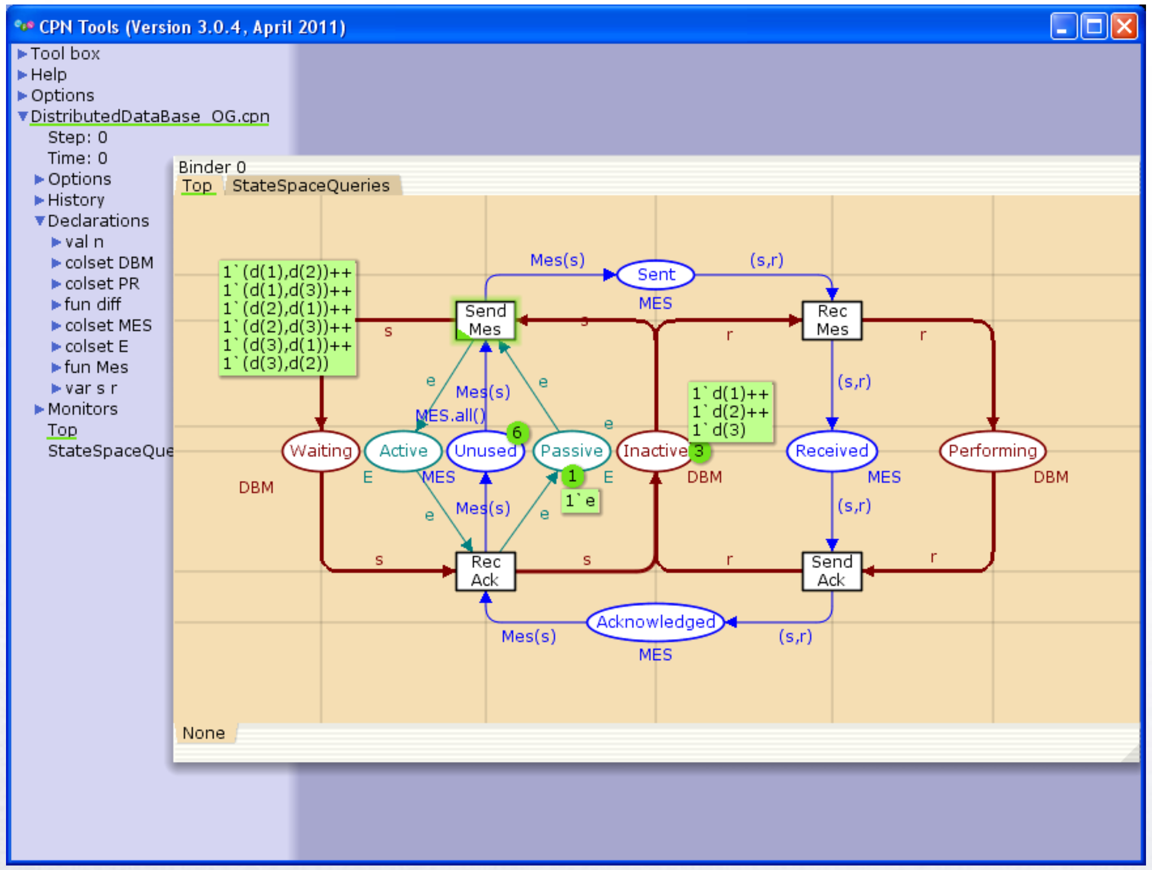
CPN Reachability Analysis

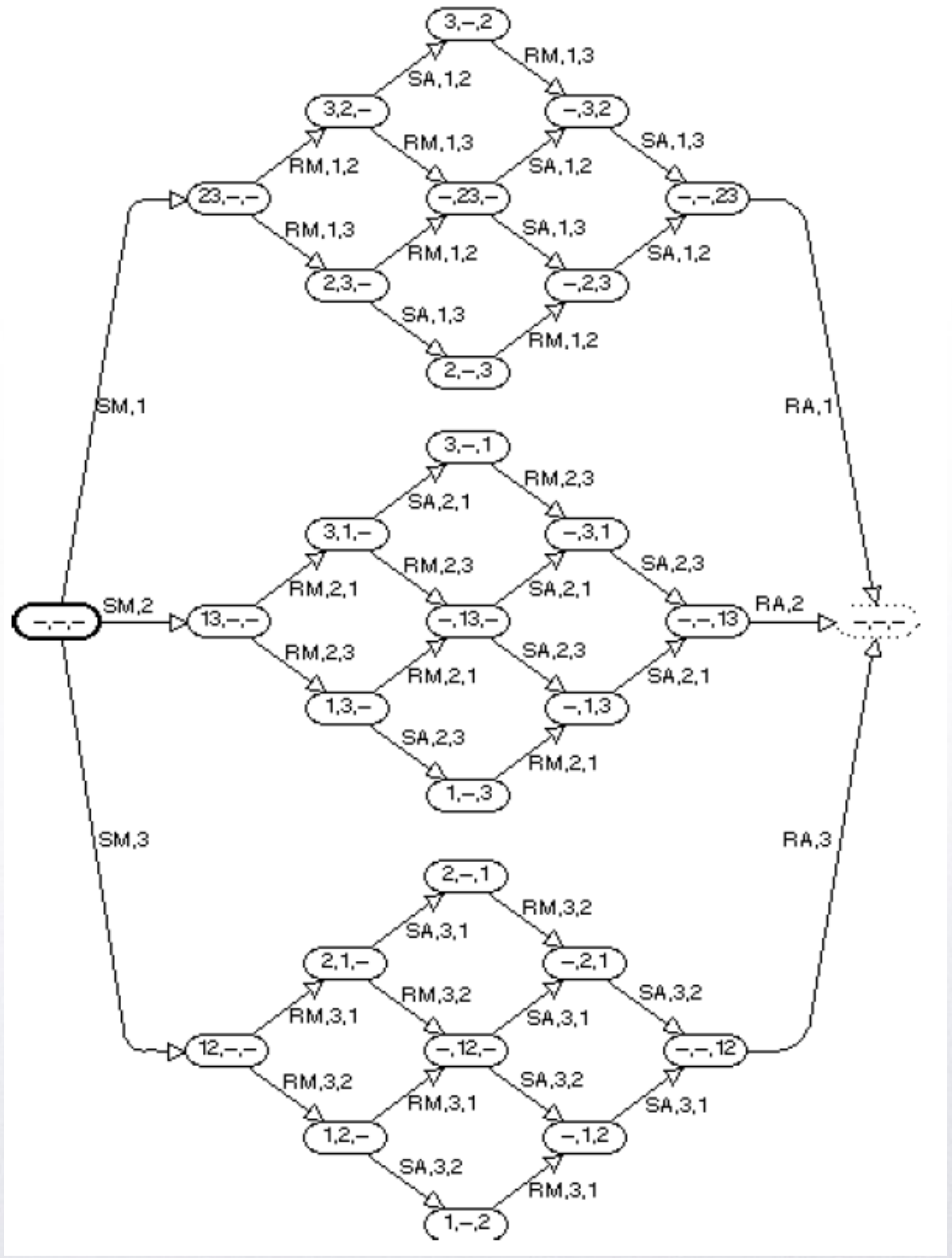


1) *boundedness*, characterising finiteness of the state space.
2) *liveness*, related to potential fireability in all reachable markings. *Deadlock-freeness* is a weaker condition in which global infinite activity (i.e. fireability) of the net system model is guaranteed, but some parts of it may not 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).

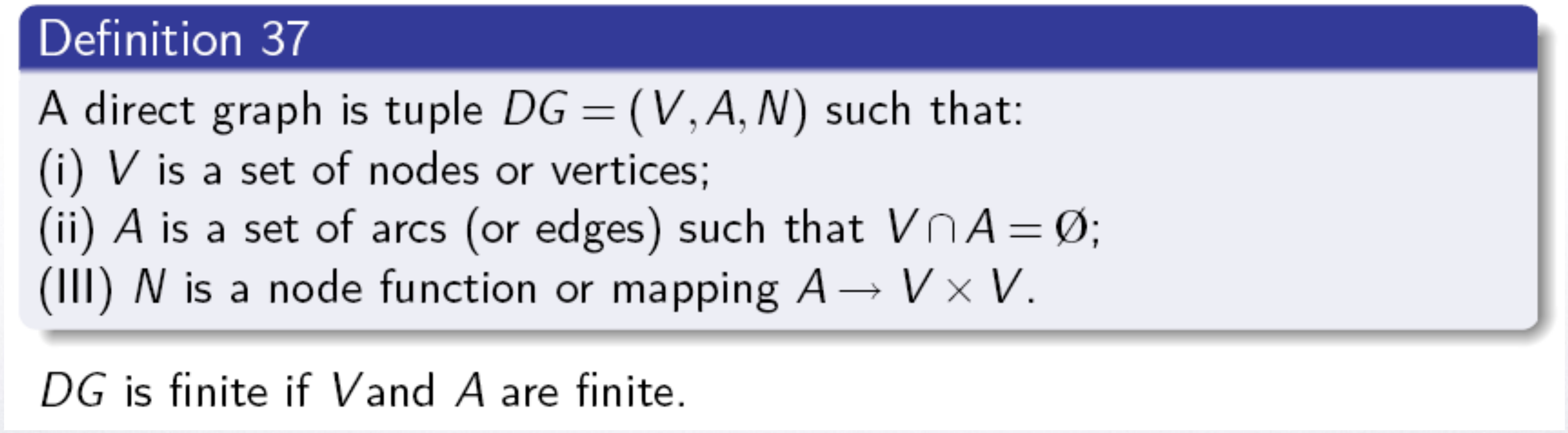
Reachability analysis in CPN is connected to building of similar "reachability graph", which is now defined over a special directed graph called **CPN state space**.

Analysis in CPN Nets





Directed Graphs

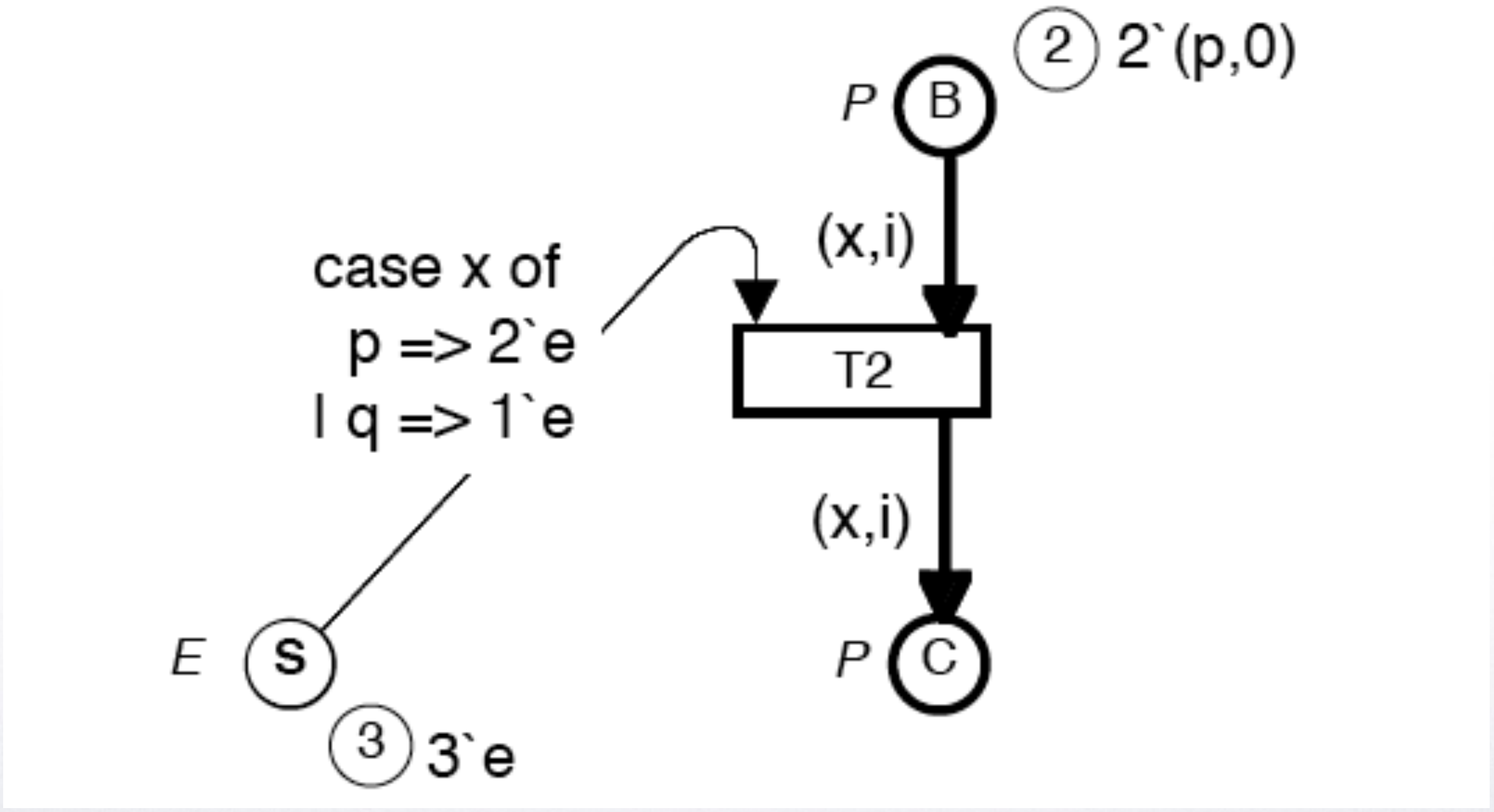


Definition 37

A direct graph is tuple $DG = (V, A, N)$ such that:
(i) V is a set of nodes or vertices;
(ii) A is a set of arcs (or edges) such that $V \cap A = \emptyset$;
(III) N is a node function or mapping $A \rightarrow V \times V$.

DG is finite if $V$and A are finite.

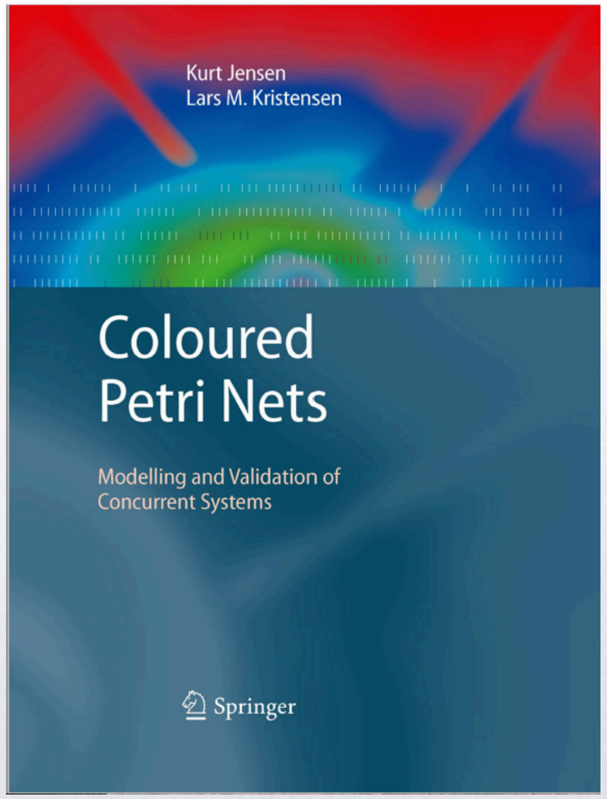
Biding



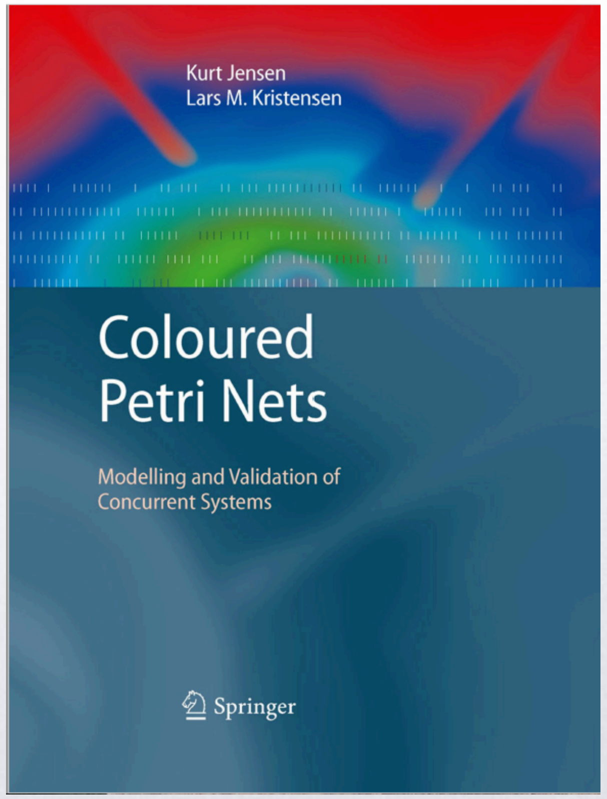
Definition 9.6. The **state space** of a Coloured Petri Net is a directed graph SS $= (N_{SS}, A_{SS})$ with arc labels from BE, where:

1. $N_{SS} = \mathscr{R}(M_0)$ is the set of **nodes**.
2. $A_{SS} = \{(M,(t,b),M') \in N_{SS} \times BE \times N_{SS} \mid M \xrightarrow{(t,b)} M'\}$ is the set of **arcs**.

SS is **finite** if and only if N_{SS} and A_{SS} are finite.

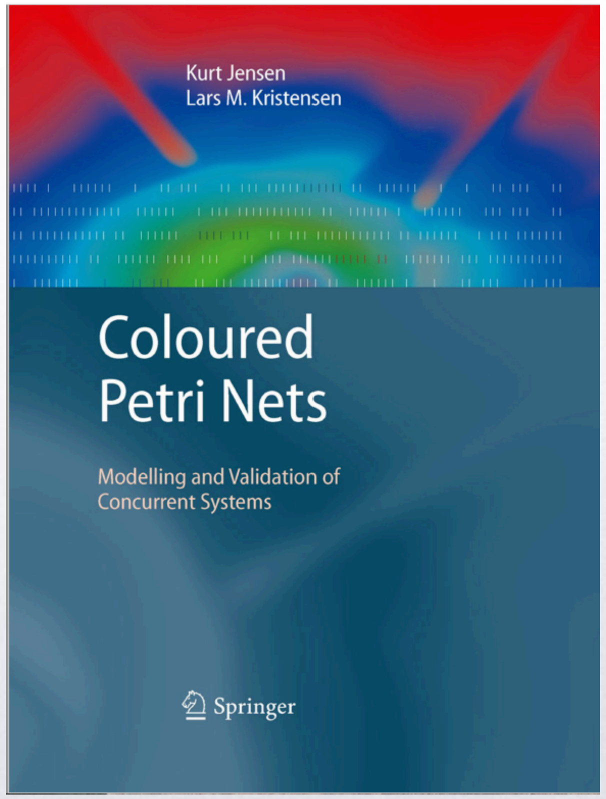


```
 1: $\text{NODES} \leftarrow \{M_0\}$
 2: $\text{UNPROCESSED} \leftarrow \{M_0\}$
 3: $\text{ARCS} \leftarrow \emptyset$
 4: while $\text{UNPROCESSED} \neq \emptyset$ do
 5:     Select a marking $M$ in UNPROCESSED
 6:     $\text{UNPROCESSED} \leftarrow \text{UNPROCESSED} - \{M\}$
 7:     for all binding elements $(t,b)$ such that $M \xrightarrow{(t,b)}$ do
 8:         Calculate $M'$ such that $M \xrightarrow{(t,b)} M'$
 9:         $\text{ARCS} \leftarrow \text{ARCS} \cup \{(M,(t,b),M')\}$
10:         if $M' \notin \text{NODES}$ then
11:             $\text{NODES} \leftarrow \text{NODES} \cup \{M'\}$
12:             $\text{UNPROCESSED} \leftarrow \text{UNPROCESSED} \cup \{M'\}$
13:         end if
14:     end for
15: end while
```

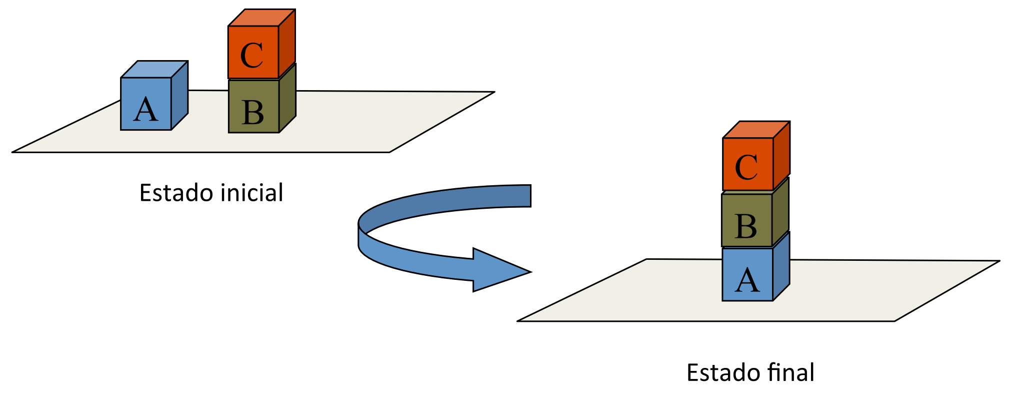


Proposition 9.7. *Let $SS = (N_{SS}, A_{SS})$ be the state space of a Coloured Petri Net CPN and let $M_1 \in \mathscr{R}(M_0)$. Then the following holds:*

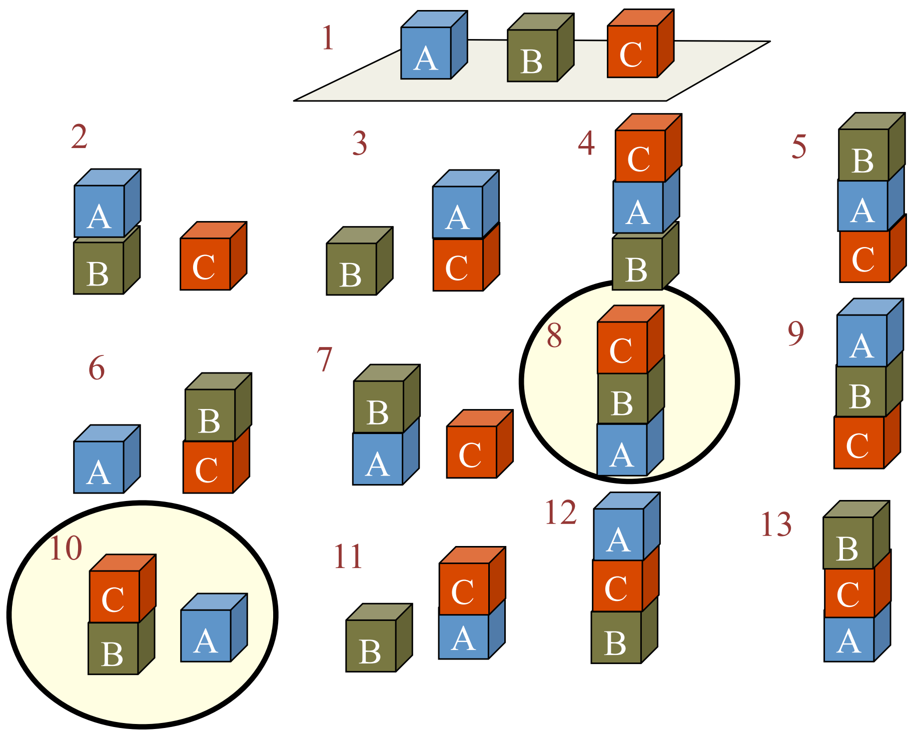
1. *$M_1 \xrightarrow{be_1} M_2 \xrightarrow{be_2} M_3 \cdots M_k \xrightarrow{be_k} M_{k+1}$ is a finite occurrence sequence of CPN if and only if $M_1\,(M_1, be_1, M_2)\,M_2\,(M_2, be_2, M_3)\,M_3 \cdots M_k\,(M_k, be_i, M_{k+1})\,M_{k+1}$ is a finite directed path in SS.*
2. *$M_1 \xrightarrow{be_1} M_2 \xrightarrow{be_2} M_3 \cdots$ is an infinite occurrence sequence of CPN if and only if $M_1\,(M_1, be_1, M_2)\,M_2\,(M_2, be_2, M_3)\,M_3 \cdots$ is a infinite directed path in SS.*

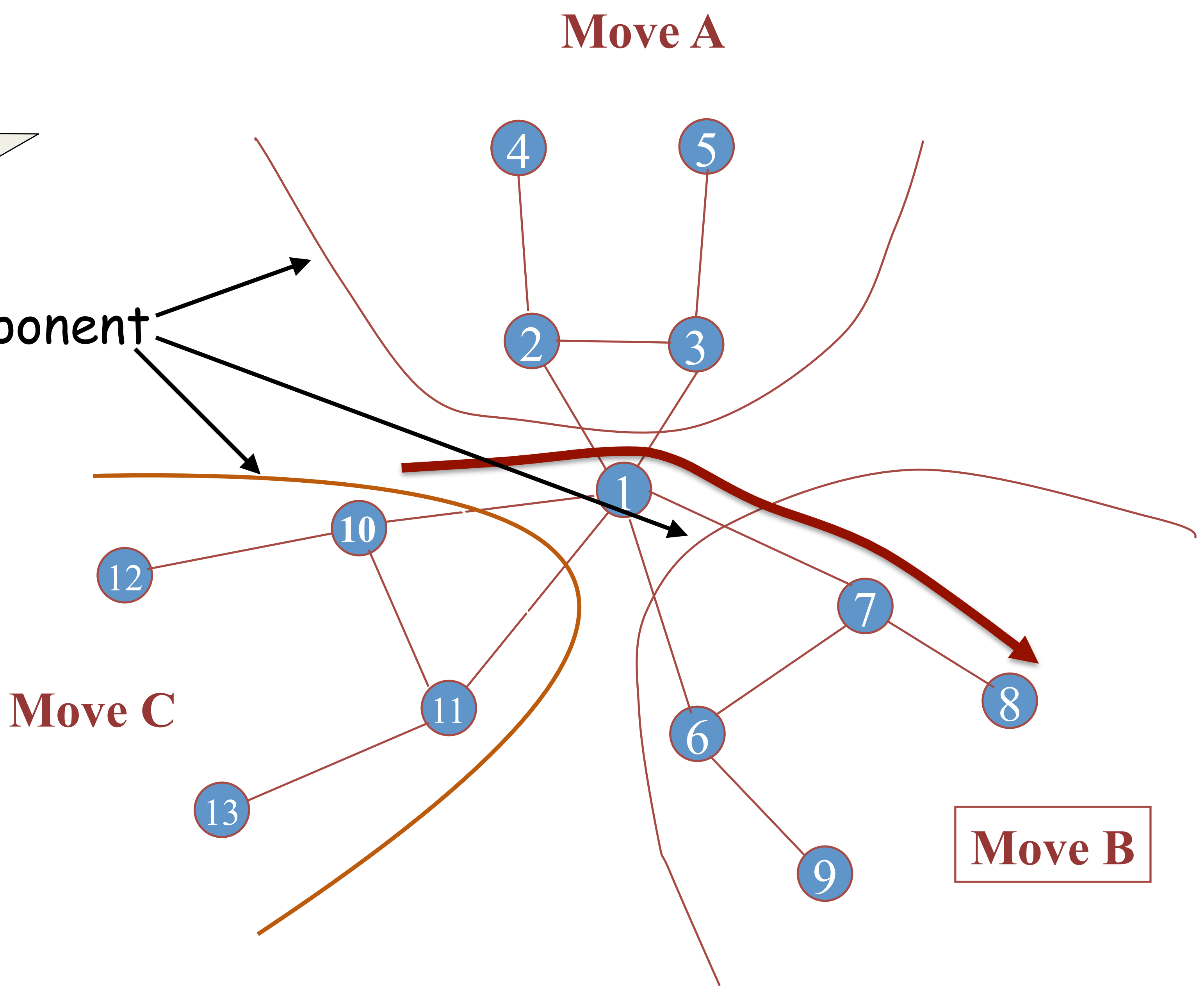


Lembrando a Aula2…



SCC-strongly connected component





Proposition 9.8. *Let $SS = (N_{SS}, A_{SS})$ be the finite state space of a Coloured Petri Net, and let $SG = (N_{SG}, A_{SG})$ be the SCC graph. Then the following holds:*

1. *A marking M' is reachable from a marking $M \in \mathscr{R}(M_0)$ if and only if*

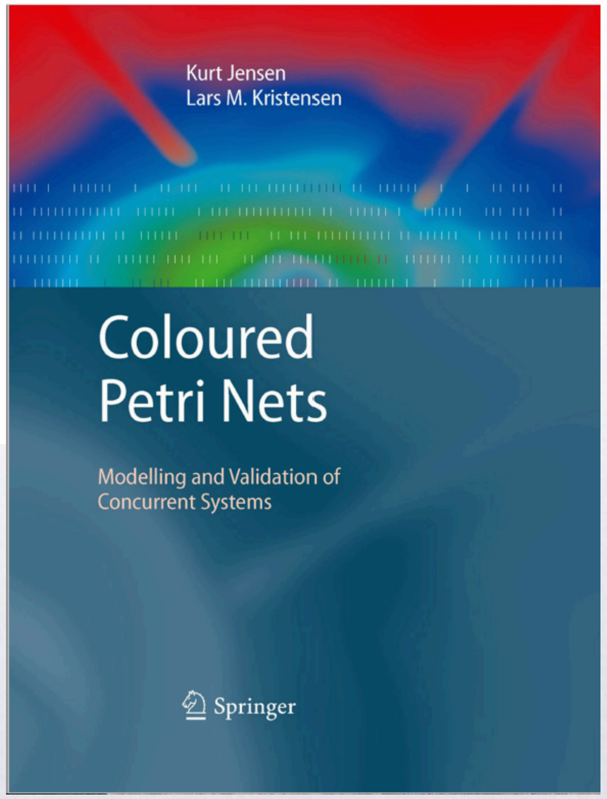
$$M' \in \mathscr{R}_{SS}(M)$$

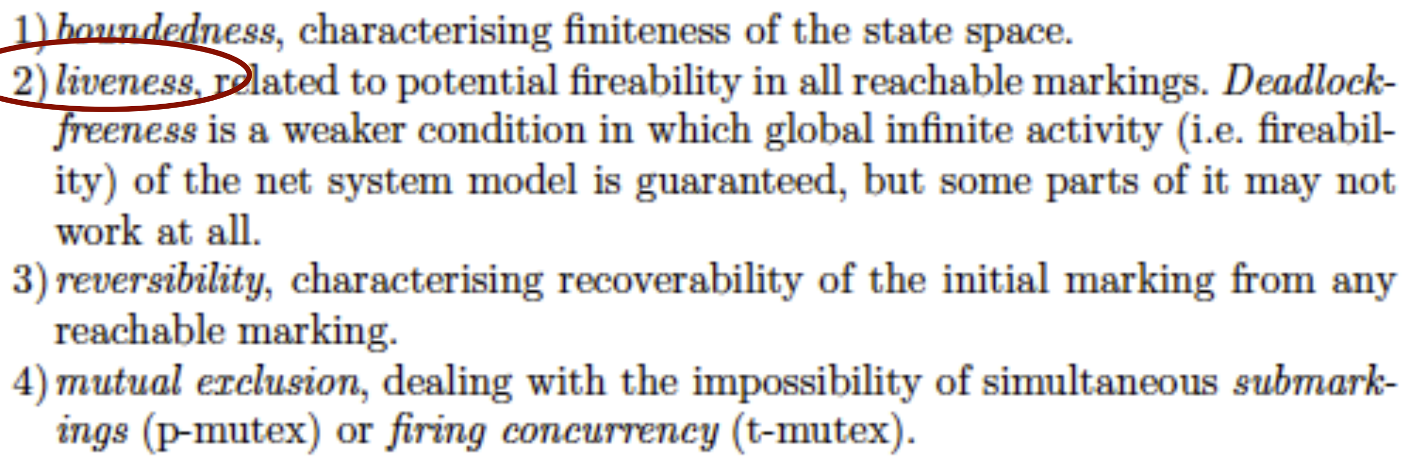
2. *A marking M' is reachable from a marking $M \in \mathscr{R}(M_0)$ if and only if*

$$SCC(M') \in \mathscr{R}_{SG}(SCC(M))$$

3. *A predicate ϕ on markings is reachable if and only if*

$$\exists M \in N_{SS} : \phi(M)$$

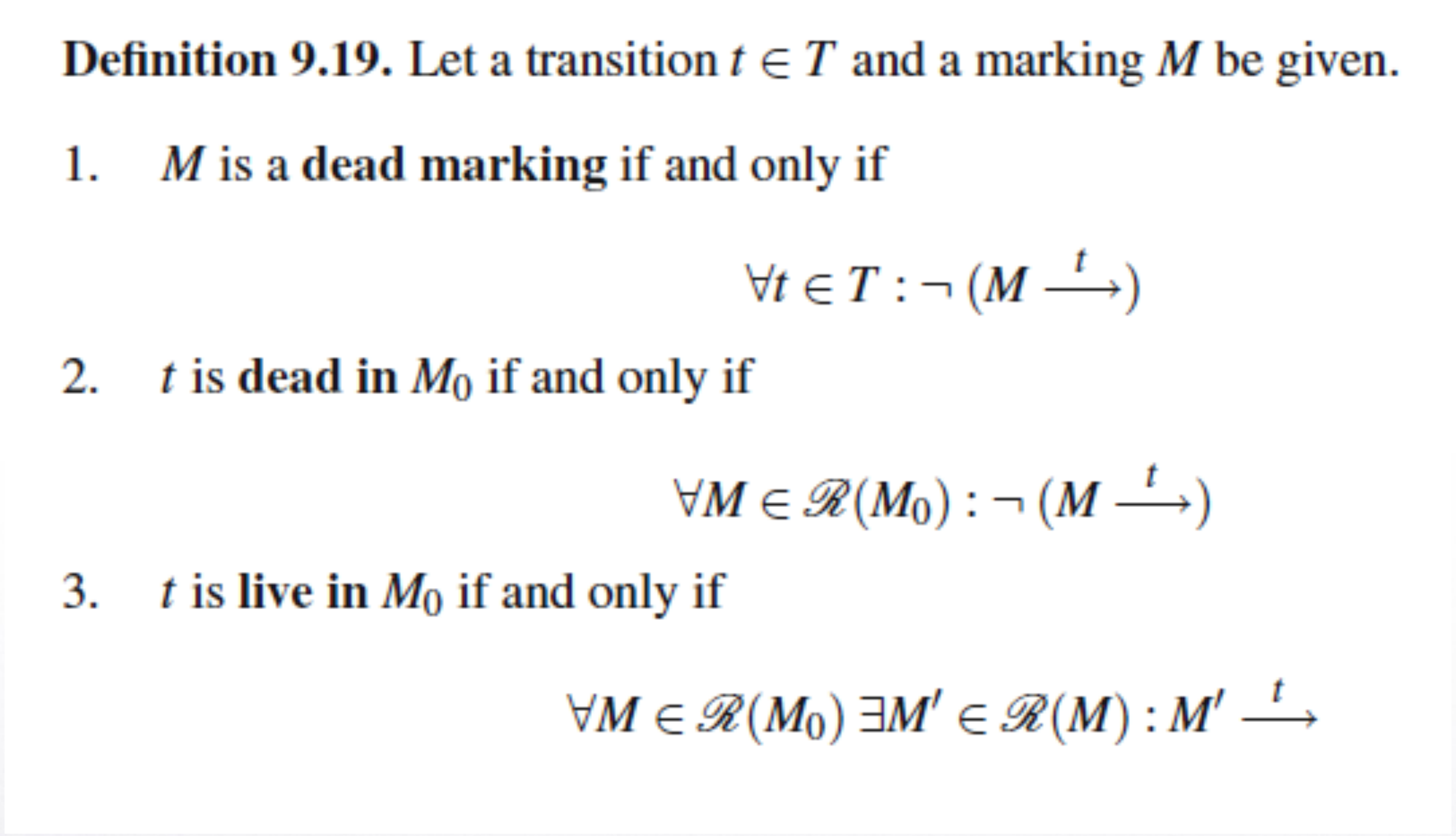

Kurt Jensen
Lars M. Kristensen
Coloured Petri Nets
Modelling and Validation of Concurrent Systems
Springer



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).

A CPN net N is said to be **live** iff $\forall M \exists | M \epsilon L(M_0)$.

A CPN net N is said to be strongly **live** iff $\forall M \exists | M \epsilon SCC(M_0)$



Definition 9.19. Let a transition $t \in T$ and a marking M be given.

1. M is a **dead marking** if and only if

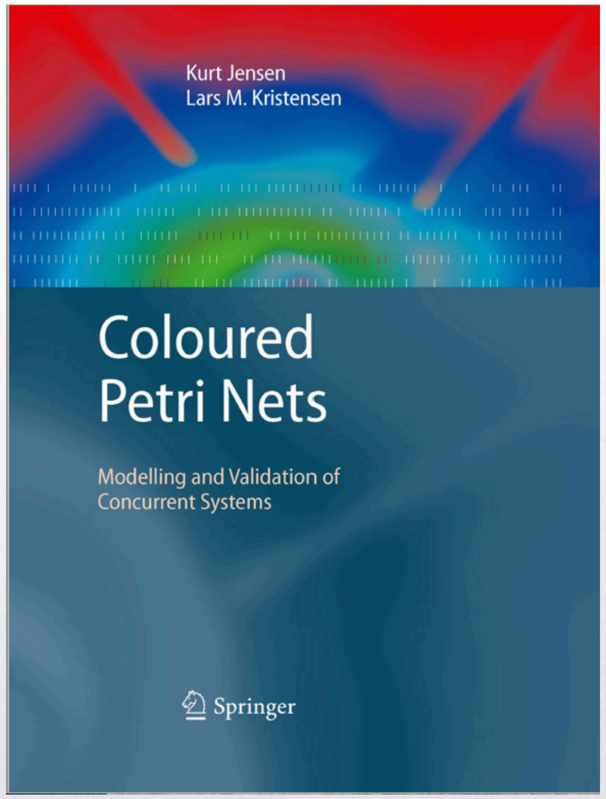
$$\forall t \in T : \neg\,(M \xrightarrow{t})$$

2. t is **dead in** M_0 if and only if

$$\forall M \in \mathscr{R}(M_0) : \neg\,(M \xrightarrow{t})$$

3. t is **live in** M_0 if and only if

$$\forall M \in \mathscr{R}(M_0)\; \exists M' \in \mathscr{R}(M) : M' \xrightarrow{t}$$



Kurt Jensen
Lars M. Kristensen

Coloured
Petri Nets

Modelling and Validation of
Concurrent Systems

Springer

Proposition 9.20. *Let $SG = (N_{SG}, A_{SG})$ be the SCC graph for the finite state space $SS = (N_{SS}, A_{SS})$ of a CP-net. Then the following holds:*

1. *A marking $M \in \mathscr{R}(M_0)$ is dead if and only if*

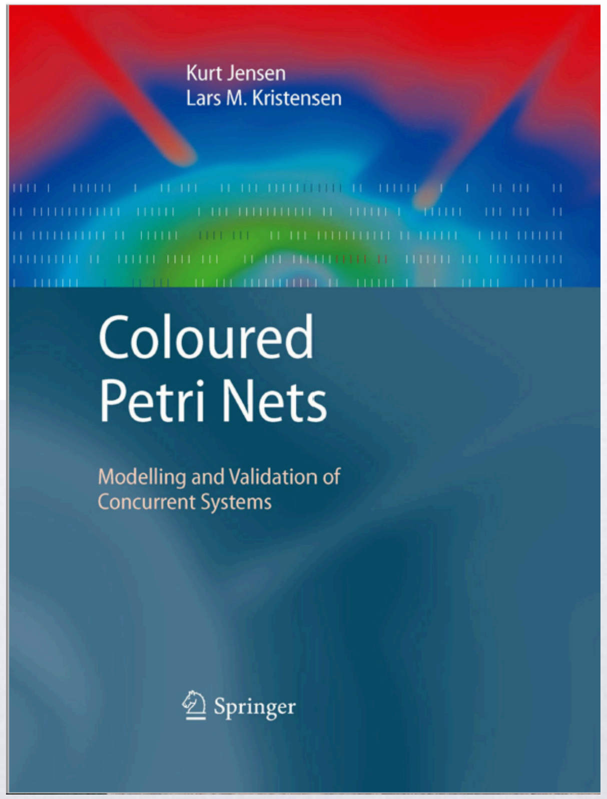
$$M \text{ is a terminal node of } SS$$

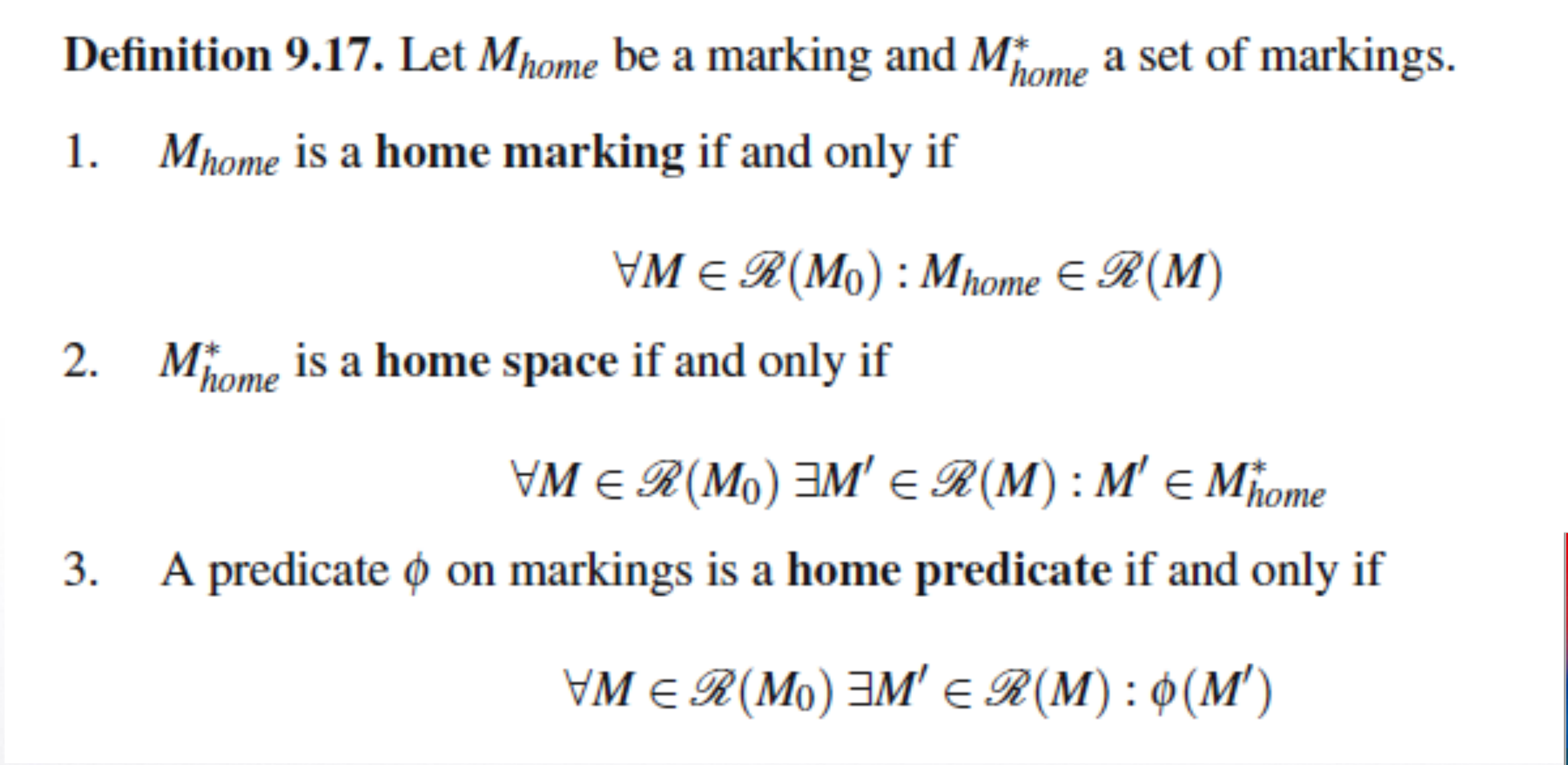
2. *A transition t is dead if and only if*

$$t \notin T(SS)$$

3. *A transition t is live if and only if*

$$\forall S \in SCC_{TM} : t \in T(S)$$





Definition 9.17. Let M_{home} be a marking and M^*_{home} a set of markings.

1. M_{home} is a **home marking** if and only if

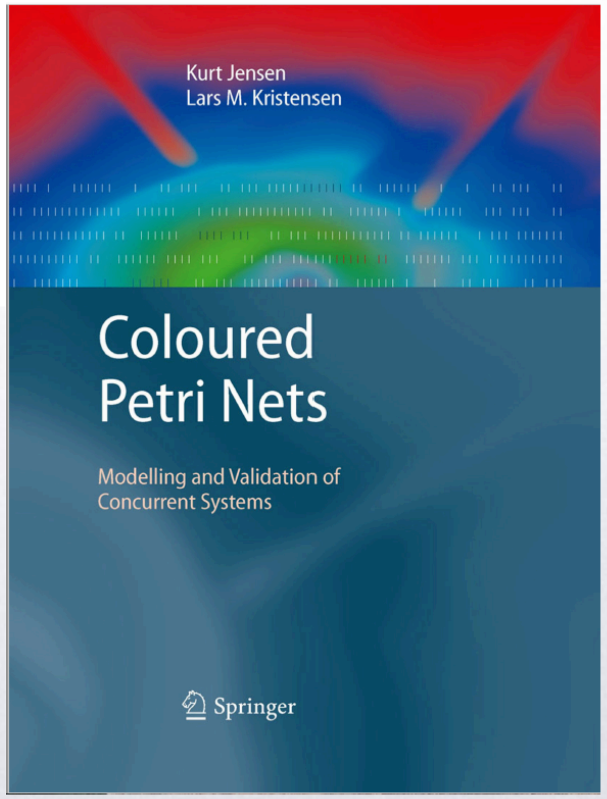
$$\forall M \in \mathscr{R}(M_0) : M_{home} \in \mathscr{R}(M)$$

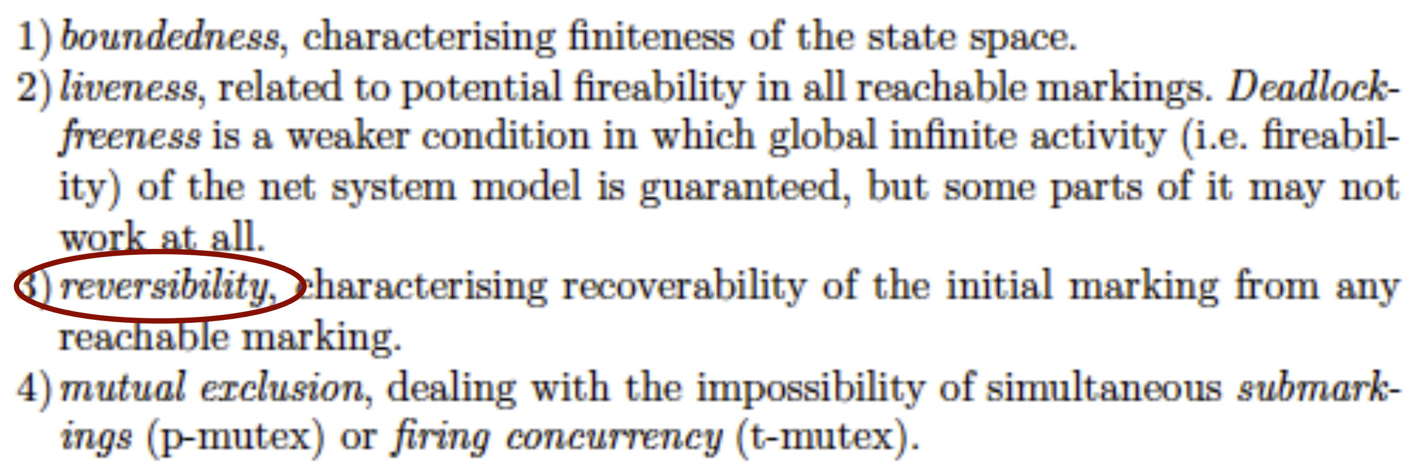
2. M^*_{home} is a **home space** if and only if

$$\forall M \in \mathscr{R}(M_0)\ \exists M' \in \mathscr{R}(M) : M' \in M^*_{home}$$

3. A predicate ϕ on markings is a **home predicate** if and only if

$$\forall M \in \mathscr{R}(M_0)\ \exists M' \in \mathscr{R}(M) : \phi(M')$$



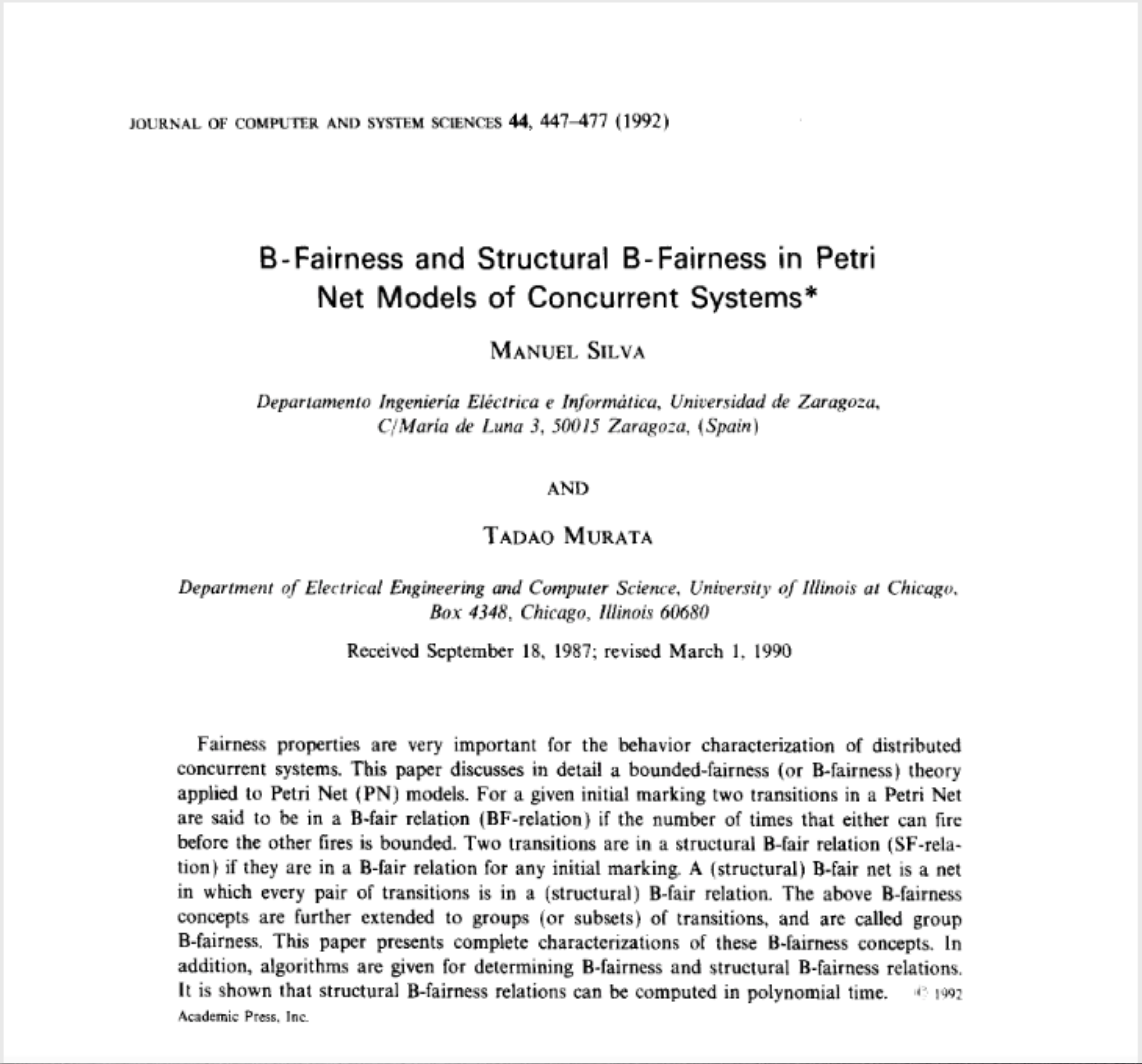


1) *boundedness*, characterising finiteness of the state space.
2) *liveness*, related to potential fireability in all reachable markings. *Deadlock-freeness* is a weaker condition in which global infinite activity (i.e. fireability) of the net system model is guaranteed, but some parts of it may not 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).

The proposition of algorithms and methods to detect home states in CPNs is still an open problem. However direct graphs can also be used to spot loops, which are essential (necessary condition) to reversibility.

Manuel Silva

Tadao Murata



JOURNAL OF COMPUTER AND SYSTEM SCIENCES **44**, 447–477 (1992)

B-Fairness and Structural B-Fairness in Petri Net Models of Concurrent Systems*

MANUEL SILVA

Departamento Ingeniería Eléctrica e Informática, Universidad de Zaragoza, C/María de Luna 3, 50015 Zaragoza, (Spain)

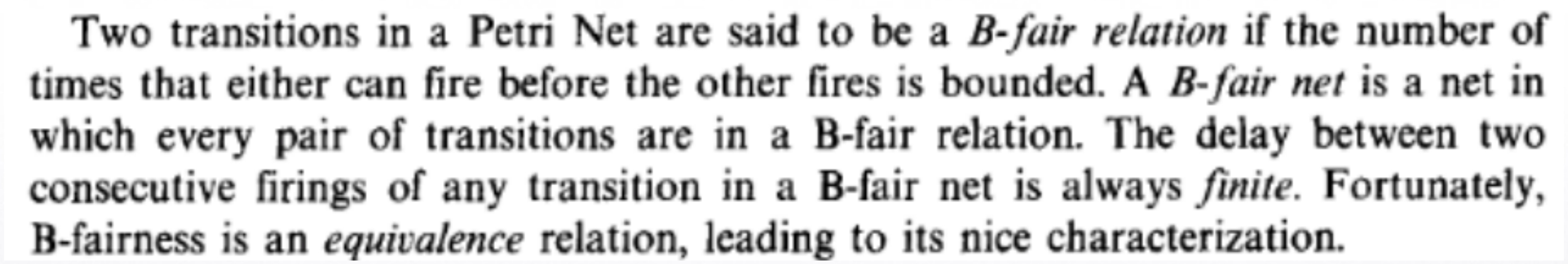
AND

TADAO MURATA

Department of Electrical Engineering and Computer Science, University of Illinois at Chicago, Box 4348, Chicago, Illinois 60680

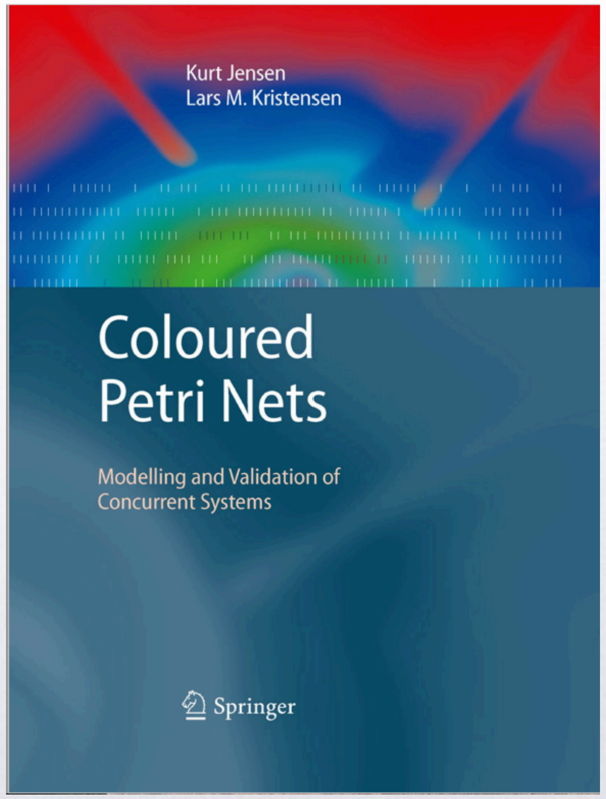
Received September 18, 1987; revised March 1, 1990

Fairness properties are very important for the behavior characterization of distributed concurrent systems. This paper discusses in detail a bounded-fairness (or B-fairness) theory applied to Petri Net (PN) models. For a given initial marking two transitions in a Petri Net are said to be in a B-fair relation (BF-relation) if the number of times that either can fire before the other fires is bounded. Two transitions are in a structural B-fair relation (SF-relation) if they are in a B-fair relation for any initial marking. A (structural) B-fair net is a net in which every pair of transitions is in a (structural) B-fair relation. The above B-fairness concepts are further extended to groups (or subsets) of transitions, and are called group B-fairness. This paper presents complete characterizations of these B-fairness concepts. In addition, algorithms are given for determining B-fairness and structural B-fairness relations. It is shown that structural B-fairness relations can be computed in polynomial time. © 1992 Academic Press, Inc.



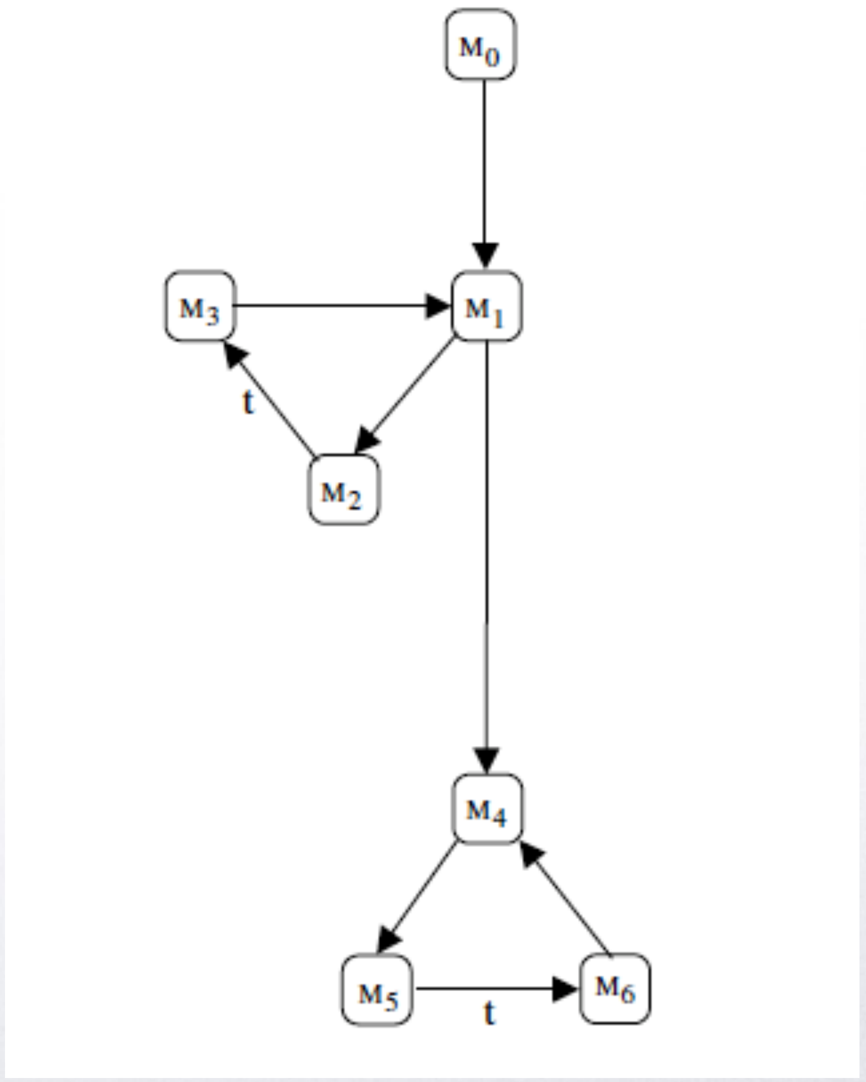
Two transitions in a Petri Net are said to be a *B-fair relation* if the number of times that either can fire before the other fires is bounded. A *B-fair net* is a net in which every pair of transitions are in a B-fair relation. The delay between two consecutive firings of any transition in a B-fair net is always *finite*. Fortunately, B-fairness is an *equivalence* relation, leading to its nice characterization.

The fairness properties give information about how often transitions occur in infinite occurrence sequences. We denote by OS_∞ the set of infinite occurrence sequences starting in the initial marking. For a transition $t \in T$ and an infinite occurrence sequence $\sigma \in OS_\infty$ we use $OC_t(\sigma)$ to denote the number of steps in which t occurs.



Impartial transitions

A transition t is said impartial if it could fire an uncountable number of times before other transitions.



DEFINITION 2.2 [9]. A Petri Net $\langle N, M_0 \rangle$ is called a *B-fair net* iff every pair of transitions are in a B-fair relation.

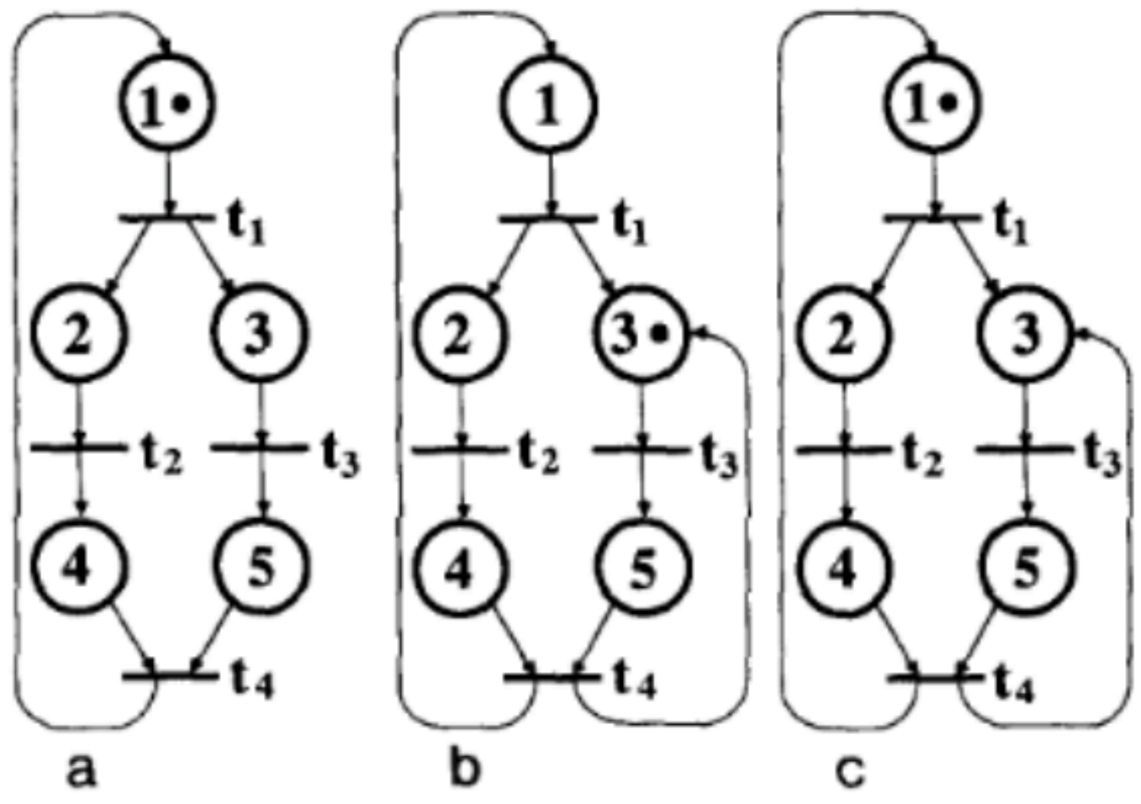
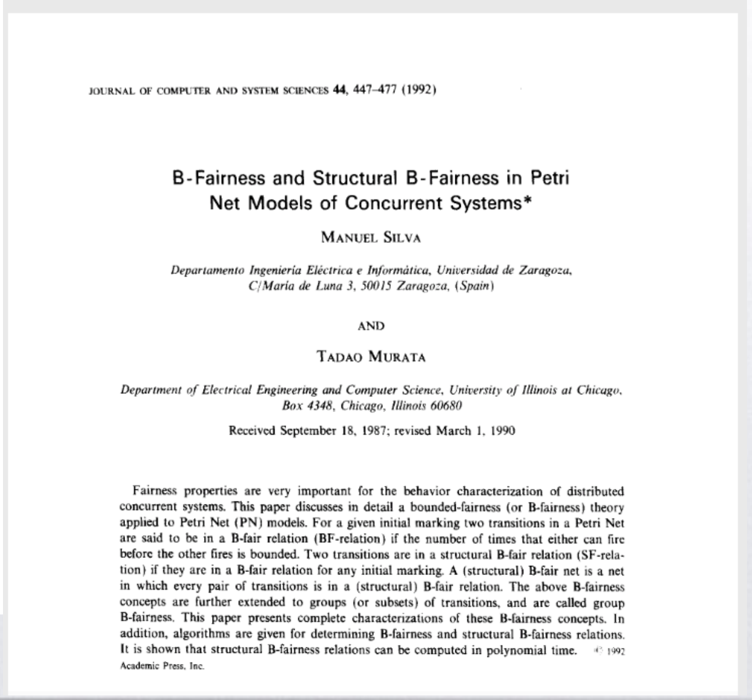


FIG. 1. B-fairness and structural B-fairness: (a) BF- and SF-net, (b) BF- and non SF-net, and (c) non-BF- and non-SF-net.



JOURNAL OF COMPUTER AND SYSTEM SCIENCES **44**, 447–477 (1992)

B-Fairness and Structural B-Fairness in Petri Net Models of Concurrent Systems*

MANUEL SILVA

Departamento Ingeniería Eléctrica e Informática, Universidad de Zaragoza, C/María de Luna 3, 50015 Zaragoza, (Spain)

AND

TADAO MURATA

Department of Electrical Engineering and Computer Science, University of Illinois at Chicago, Box 4348, Chicago, Illinois 60680

Received September 18, 1987; revised March 1, 1990

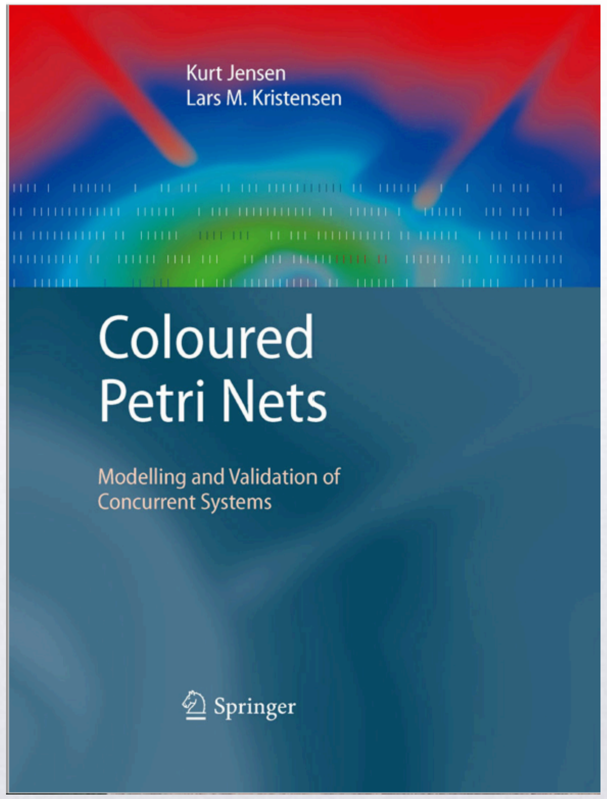
Fairness properties are very important for the behavior characterization of distributed concurrent systems. This paper discusses in detail a bounded-fairness (or B-fairness) theory applied to Petri Net (PN) models. For a given initial marking two transitions in a Petri Net are said to be in a B-fair relation (BF-relation) if the number of times that either can fire before the other fires is bounded. Two transitions are in a structural B-fair relation (SF-relation) if they are in a B-fair relation for any initial marking. A (structural) B-fair net is a net in which every pair of transitions is in a (structural) B-fair relation. The above B-fairness concepts are further extended to groups (or subsets) of transitions, and are called group B-fairness. This paper presents complete characterizations of these B-fairness concepts. In addition, algorithms are given for determining B-fairness and structural B-fairness relations. It is shown that structural B-fairness relations can be computed in polynomial time. © 1992 Academic Press, Inc.

1. (t,b) is **impartial** if and only if

$$\forall \sigma \in OS_\infty : OC_{(t,b)}(\sigma) = \infty$$

2. X is **impartial** if and only if

$$\forall \sigma \in OS_\infty : OC_X(\sigma) = \infty$$

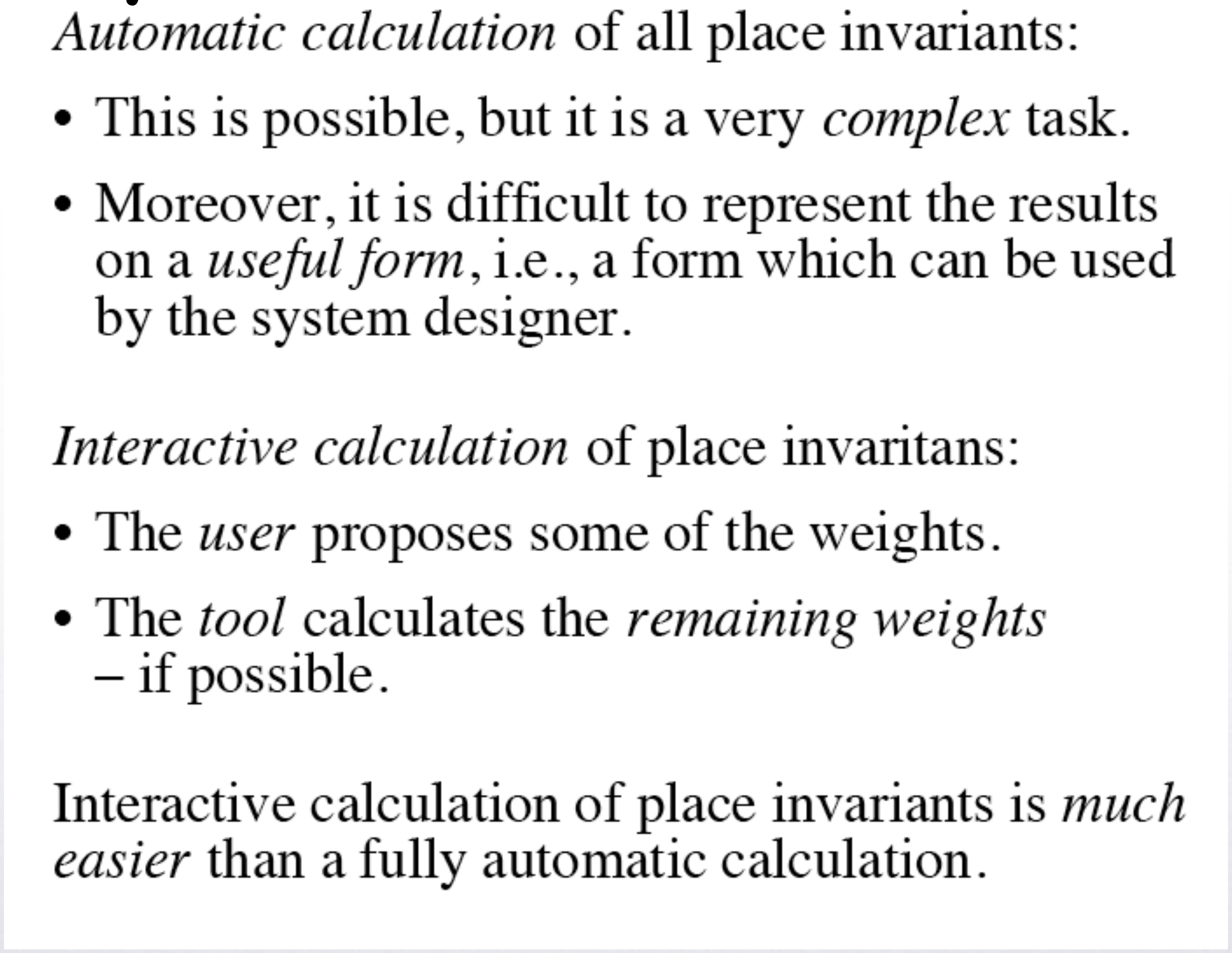

Kurt Jensen
Lars M. Kristensen
Coloured Petri Nets
Modelling and Validation of Concurrent Systems
Springer

Invariants

Generally speaking, an **invariant** is a quantity that remains constant during the execution of a given algorithm. In other words, none of the allowed operations changes the value of the invariant. The invariant principle is extremely useful in analyzing the end result (or possible end results) of an algorithm, because we can discard any potential result that has a different value for the invariant as impossible to reach.

Alexander Katz, Pi Han Goh, Geoff Pilling

Automating the invariant analysis



Automatic calculation of all place invariants:

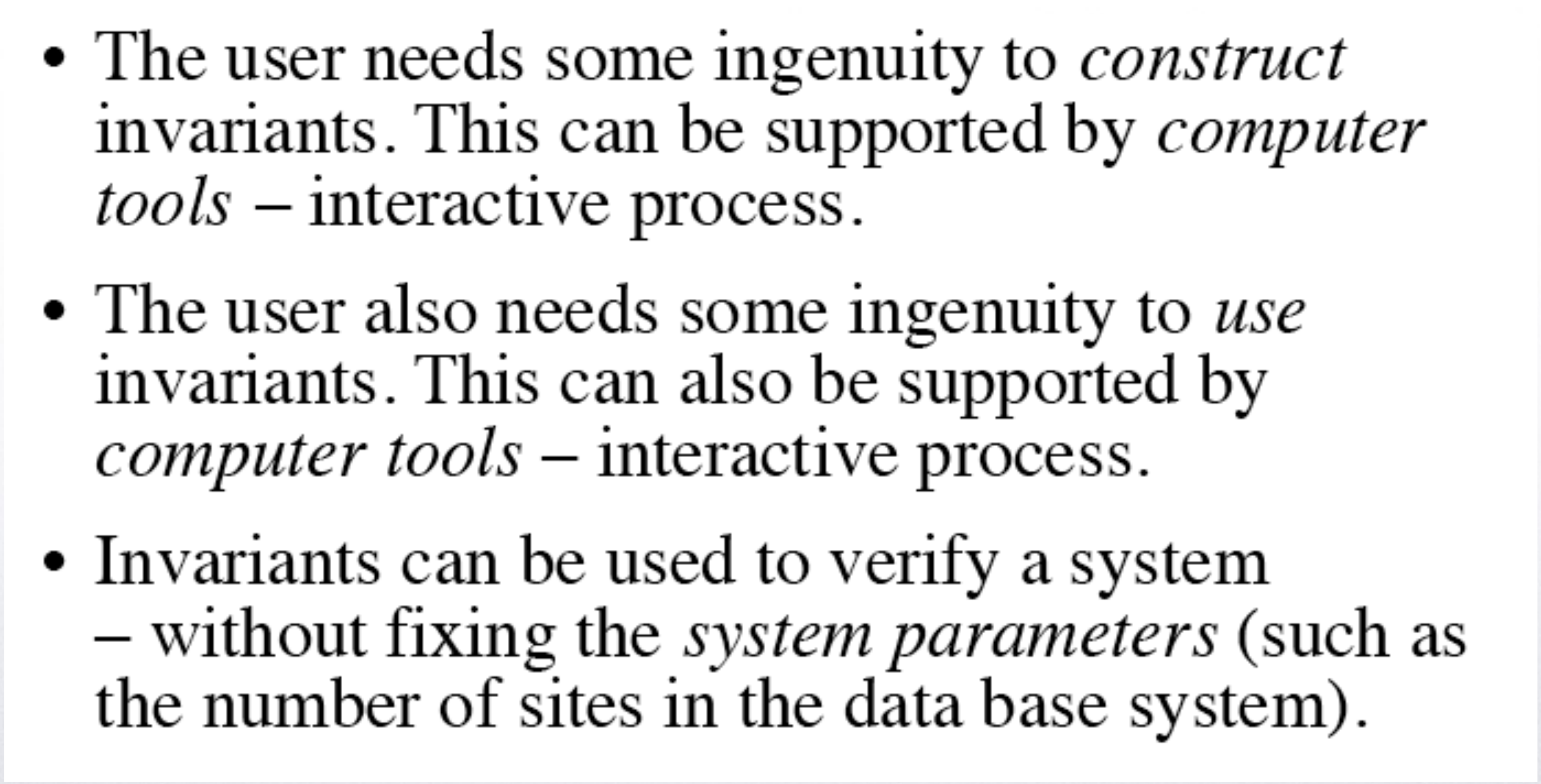
- This is possible, but it is a very *complex* task.
- Moreover, it is difficult to represent the results on a *useful form*, i.e., a form which can be used by the system designer.

Interactive calculation of place invaritans:

- The *user* proposes some of the weights.
- The *tool* calculates the *remaining weights* – if possible.

Interactive calculation of place invariants is *much easier* than a fully automatic calculation.

The invariant method in CPN



- The user needs some ingenuity to *construct* invariants. This can be supported by *computer tools* – interactive process.
- The user also needs some ingenuity to *use* invariants. This can also be supported by *computer tools* – interactive process.
- Invariants can be used to verify a system – without fixing the *system parameters* (such as the number of sites in the data base system).

Invariants are a very important feature in CPN Design. However, we should not expect to solve the design problem by just inserting invariant analysis.

Besides those inherent problems with invariants, the difficulty to apply this approach to large systems is still present.

Overview

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

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

Redes de Petri

Redes Clássicas

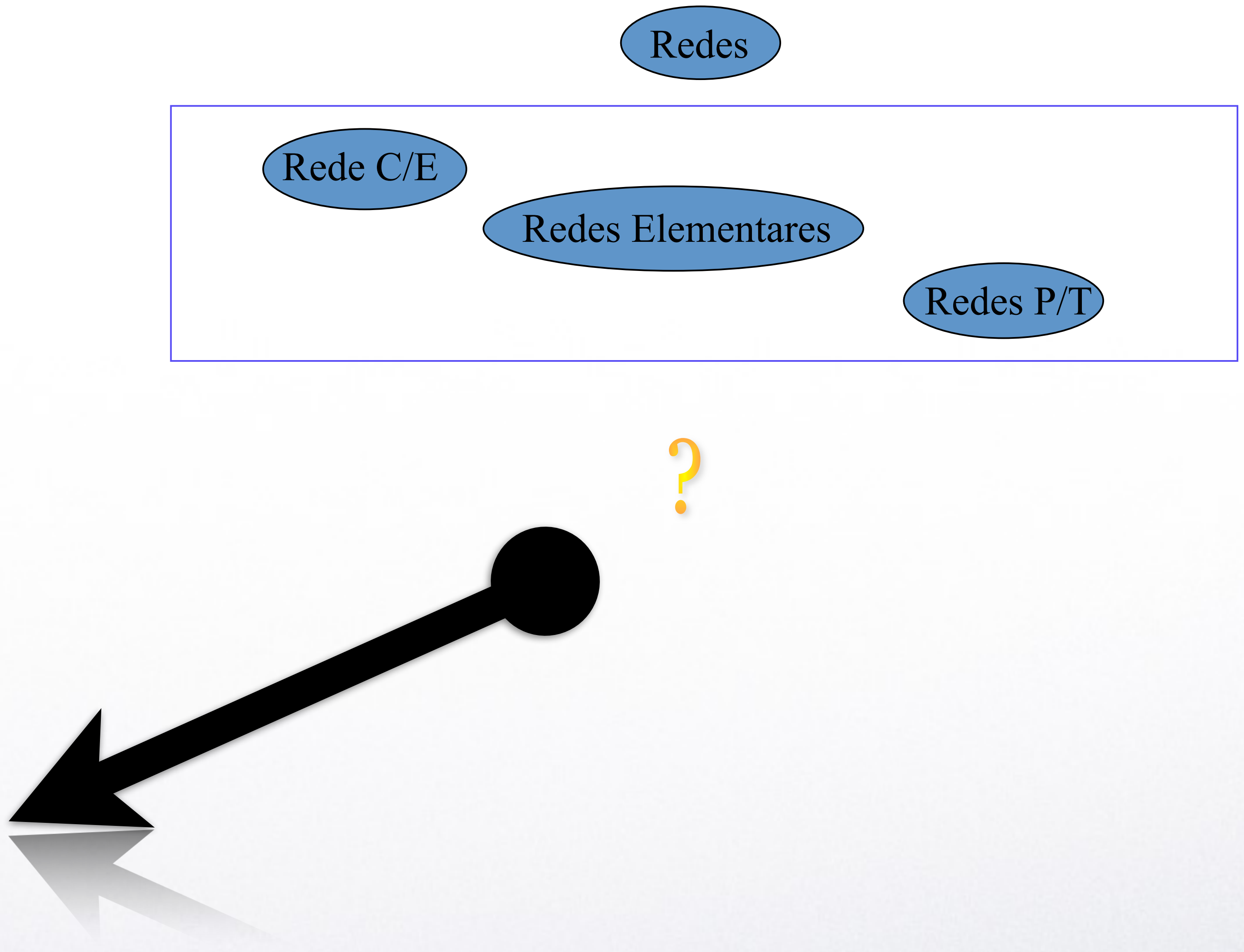
Redes P/T (seriam o padrão)

Redes de Alto Nível

(HLPNs, Redes Coloridas, etc)

Redes Estendidas

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



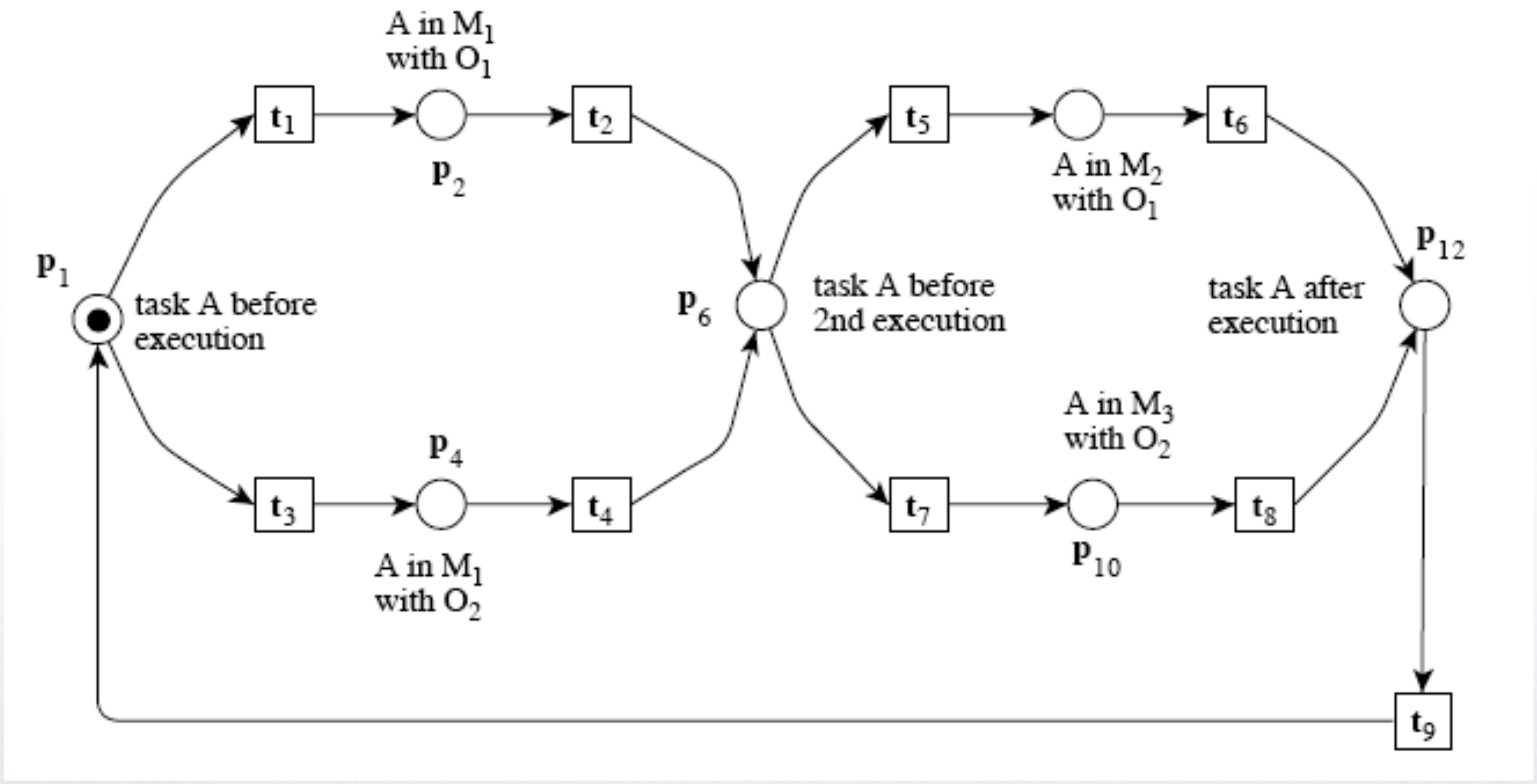
Modelagem de Sistemas Discretos

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

Procedimento de modelagem e análise

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

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



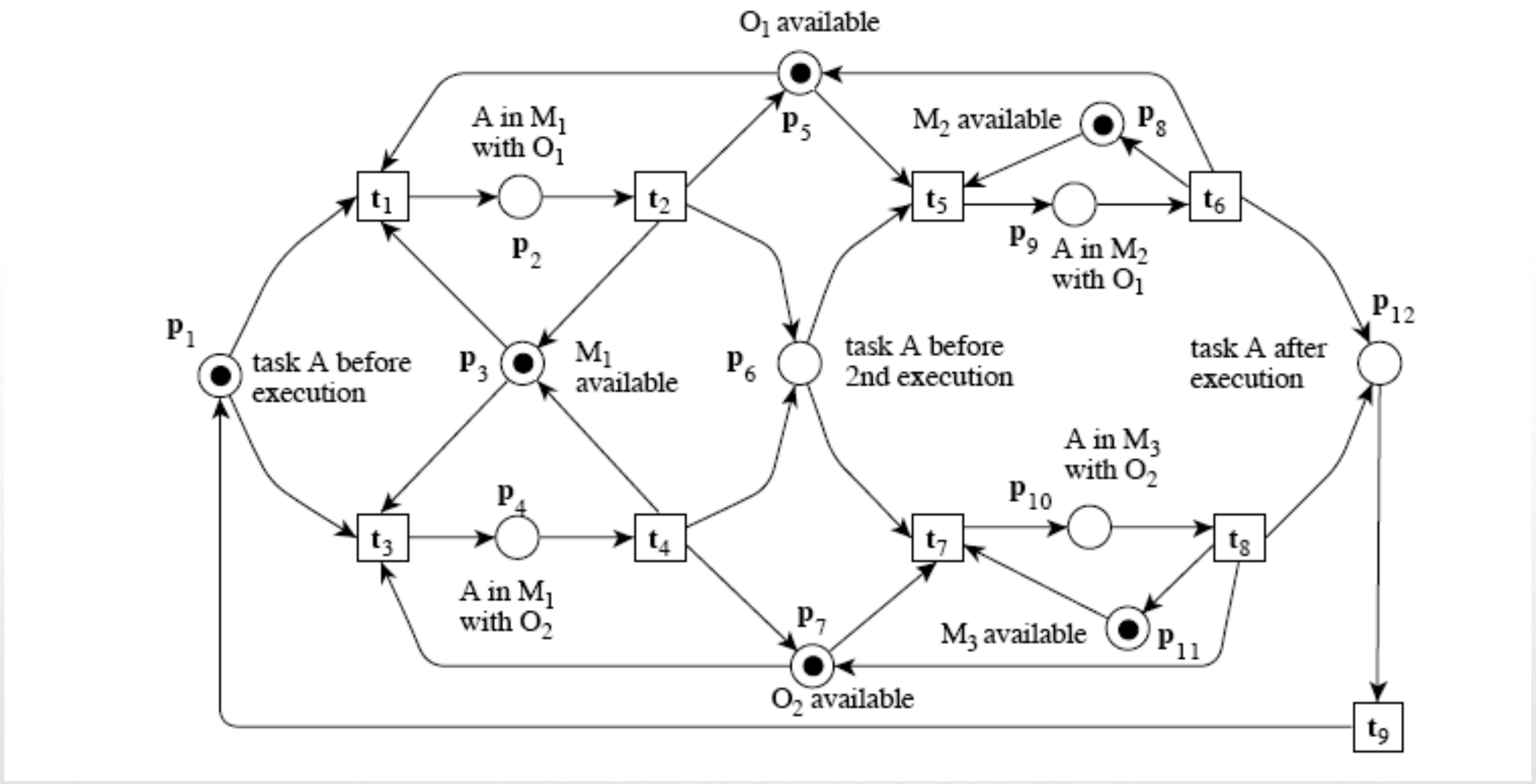
Girauld, C. and Valk, R.; Petri Nets for Systems Engineering, Springer-Verlag, 2003

Análise

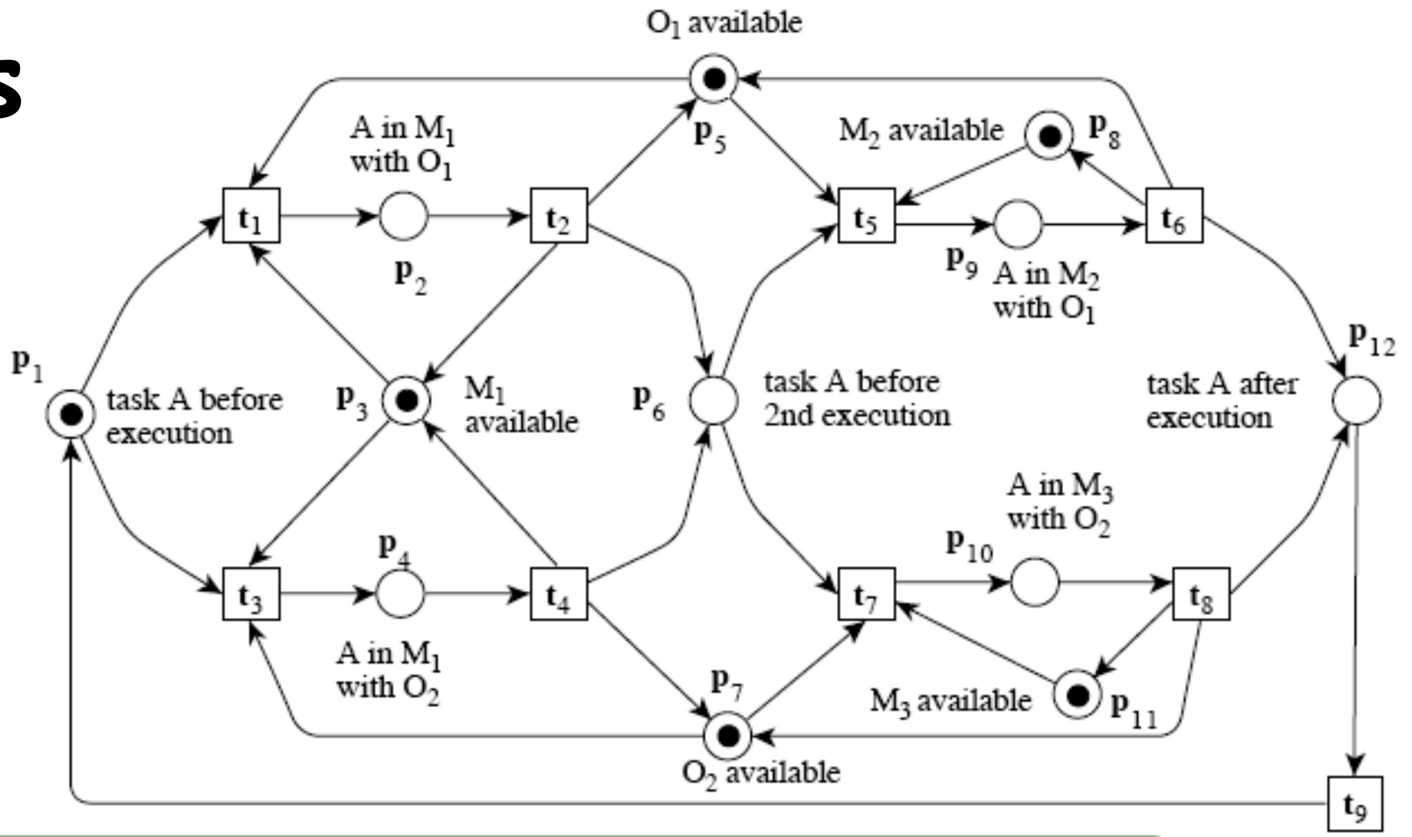
O sistema é cíclico, e permite a combinação das operações em qualquer ordem (e portanto o sistema estaria preparado para implementar qualquer receita de peça). O sistema é conservativo, de modo que cada peça seria representada por uma marca que deve estar em algum dos lugares já especificados.

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

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



Análise de Invariantes



Análise de invariantes

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

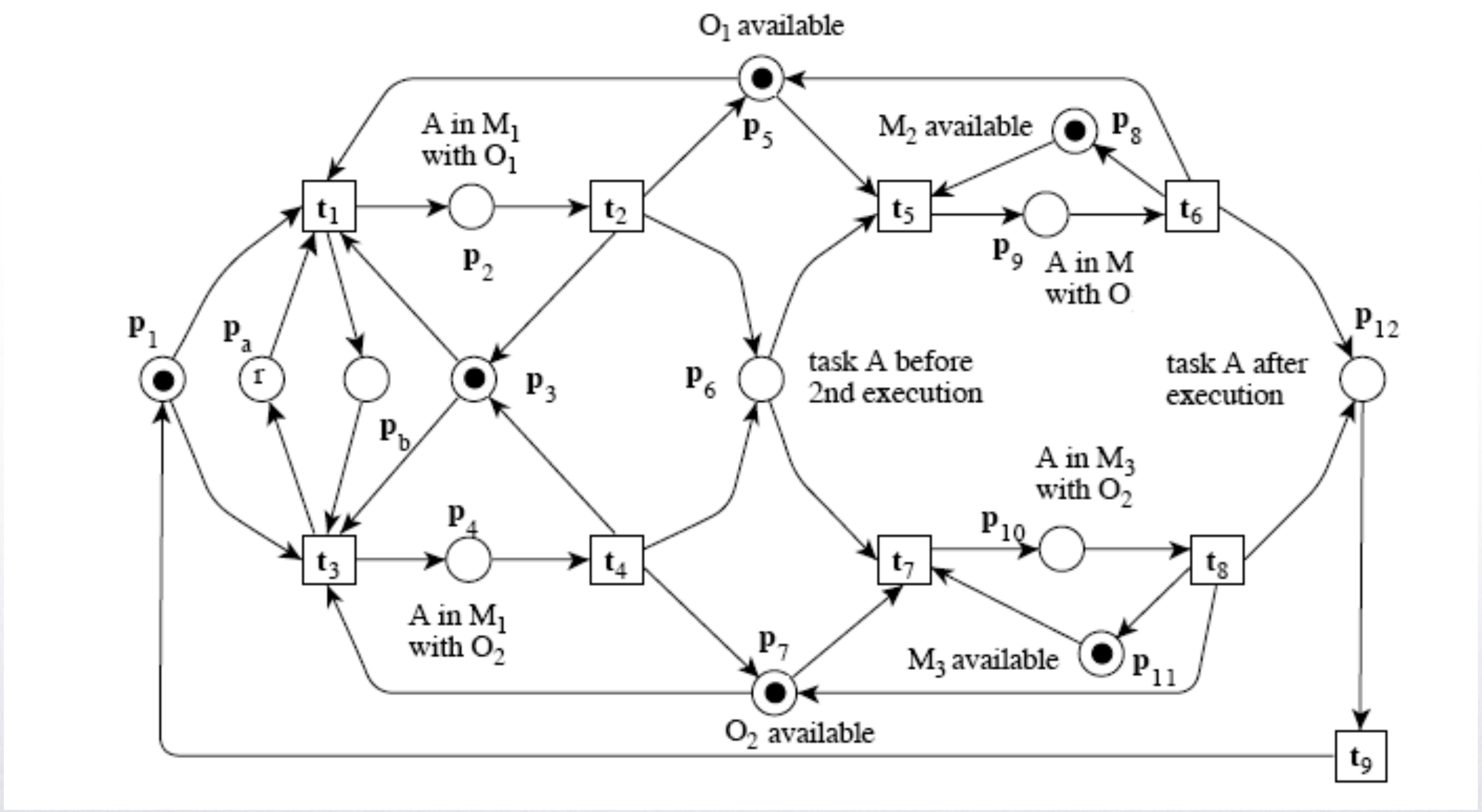
i) $m[p_2] + m[p_5] + m[p_9] = 1;$

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

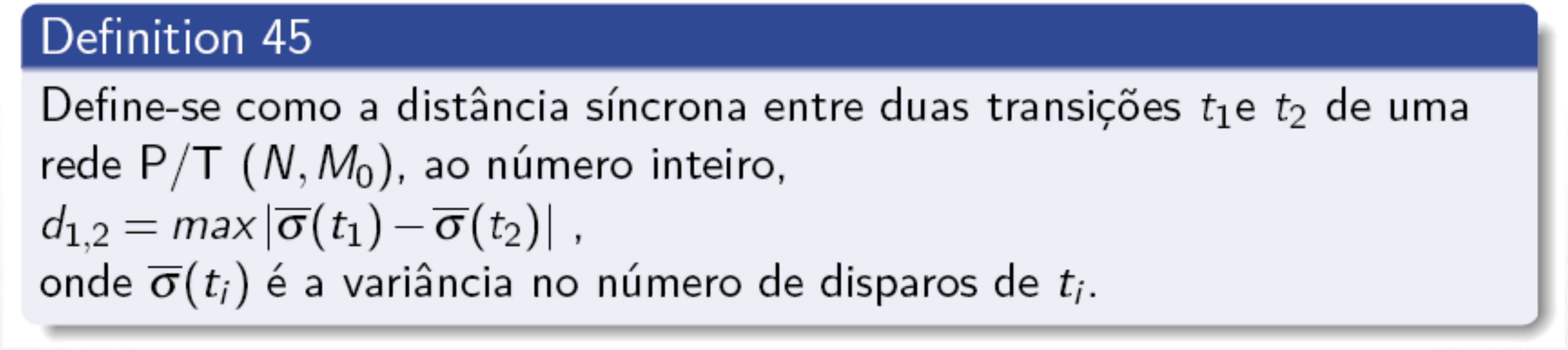
iii) $m[p_2] + m[p_3] + m[p_4] = 1;$

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

Introduzindo Sincronização



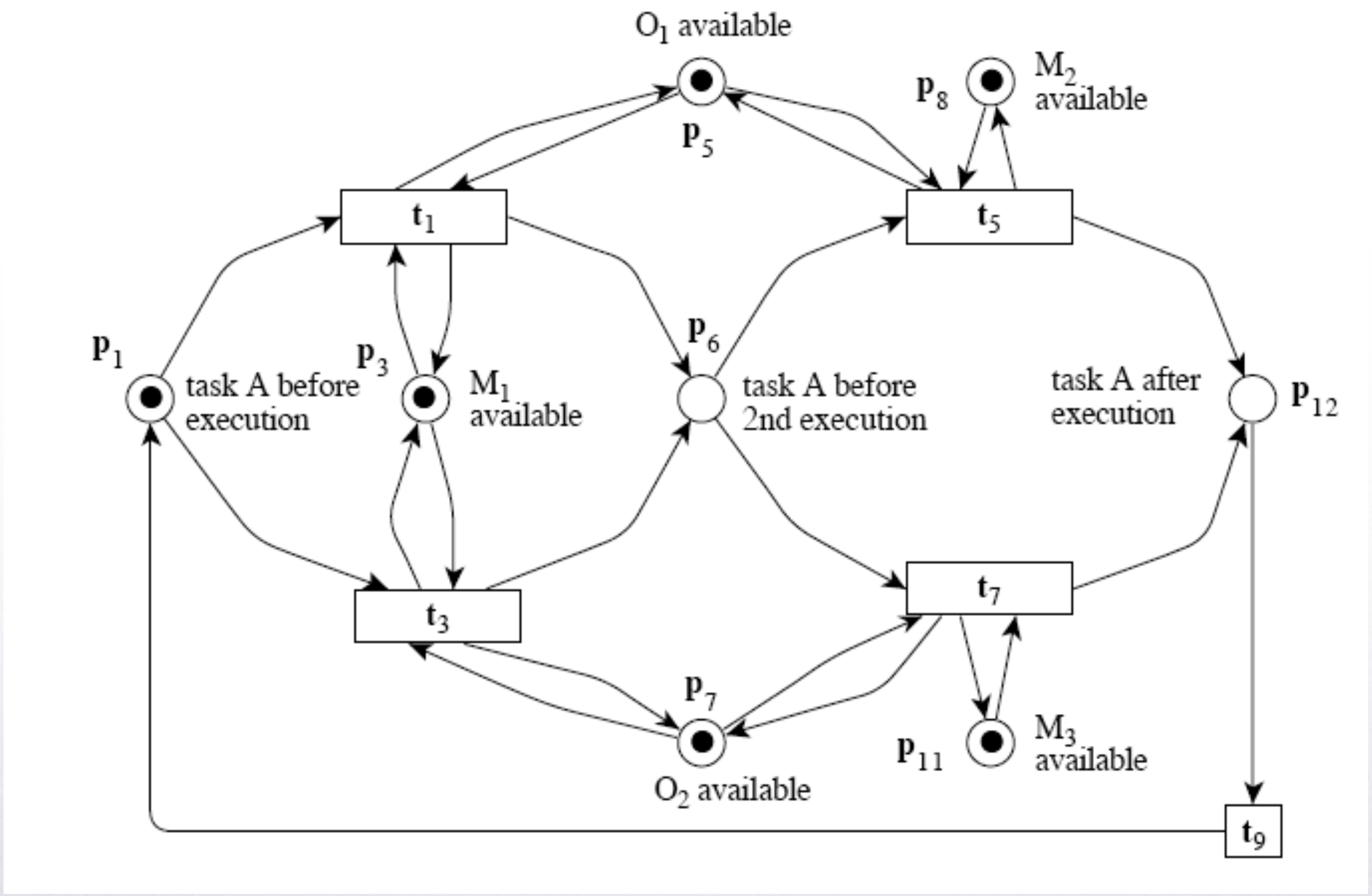
Distância Síncrona



Definition 45

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

Reduções



O1 available
p5
p8
M2 available
t1
t5
p1
task A before execution
p3
M1 available
p6
task A before 2nd execution
task A after execution
p12
t3
t7
p7
p11
M3 available
O2 available
t9

Buscando um processo de projeto

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

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

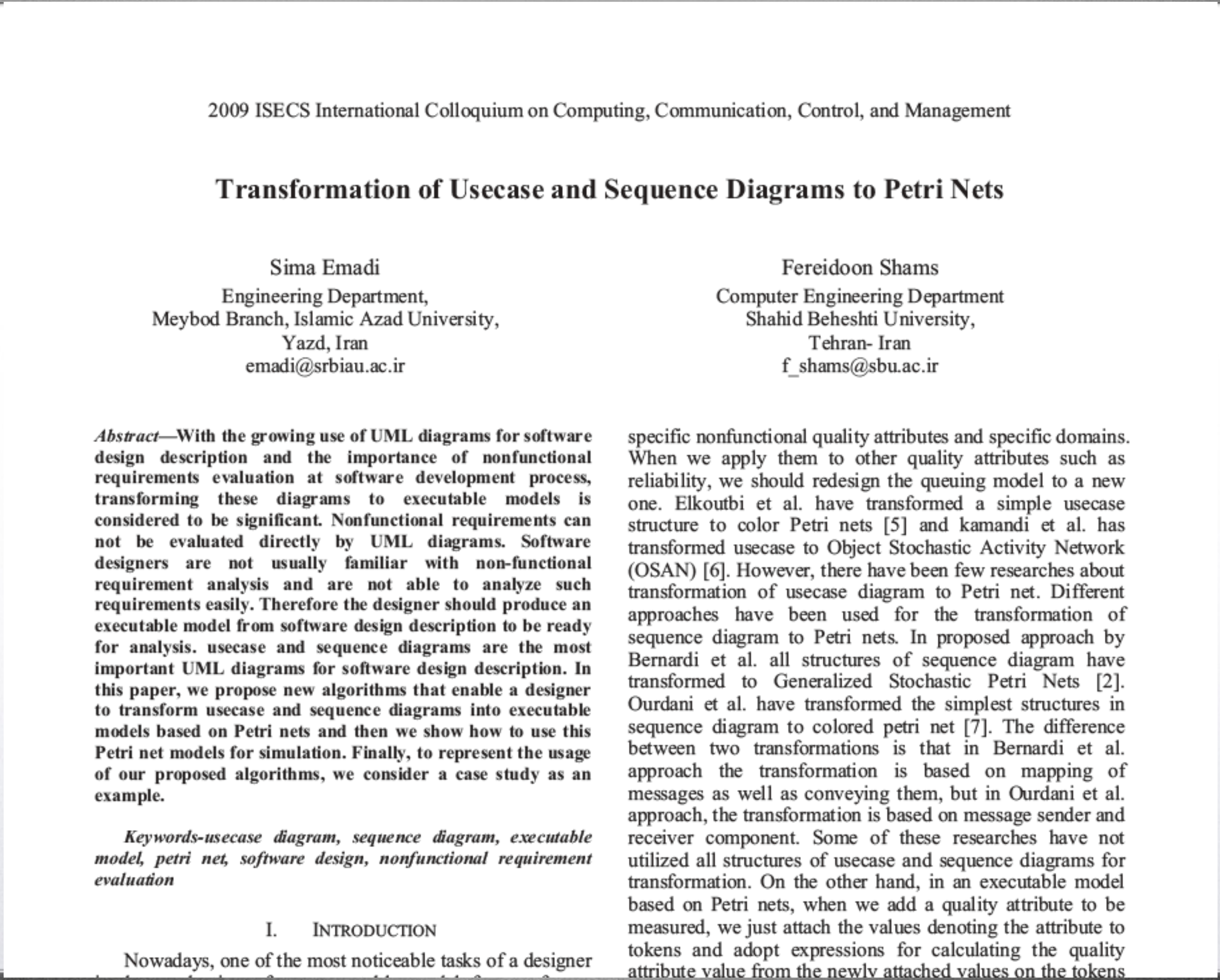
Requisitos: o início de um grande problema

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

Análise de Requisitos, Síntese de redes, Building blocks

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

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



2009 ISECS International Colloquium on Computing, Communication, Control, and Management

Transformation of Usecase and Sequence Diagrams to Petri Nets

Sima Emadi
Engineering Department,
Meybod Branch, Islamic Azad University,
Yazd, Iran
emadi@srbiau.ac.ir

Fereidoon Shams
Computer Engineering Department
Shahid Beheshti University,
Tehran- Iran
f_shams@sbu.ac.ir

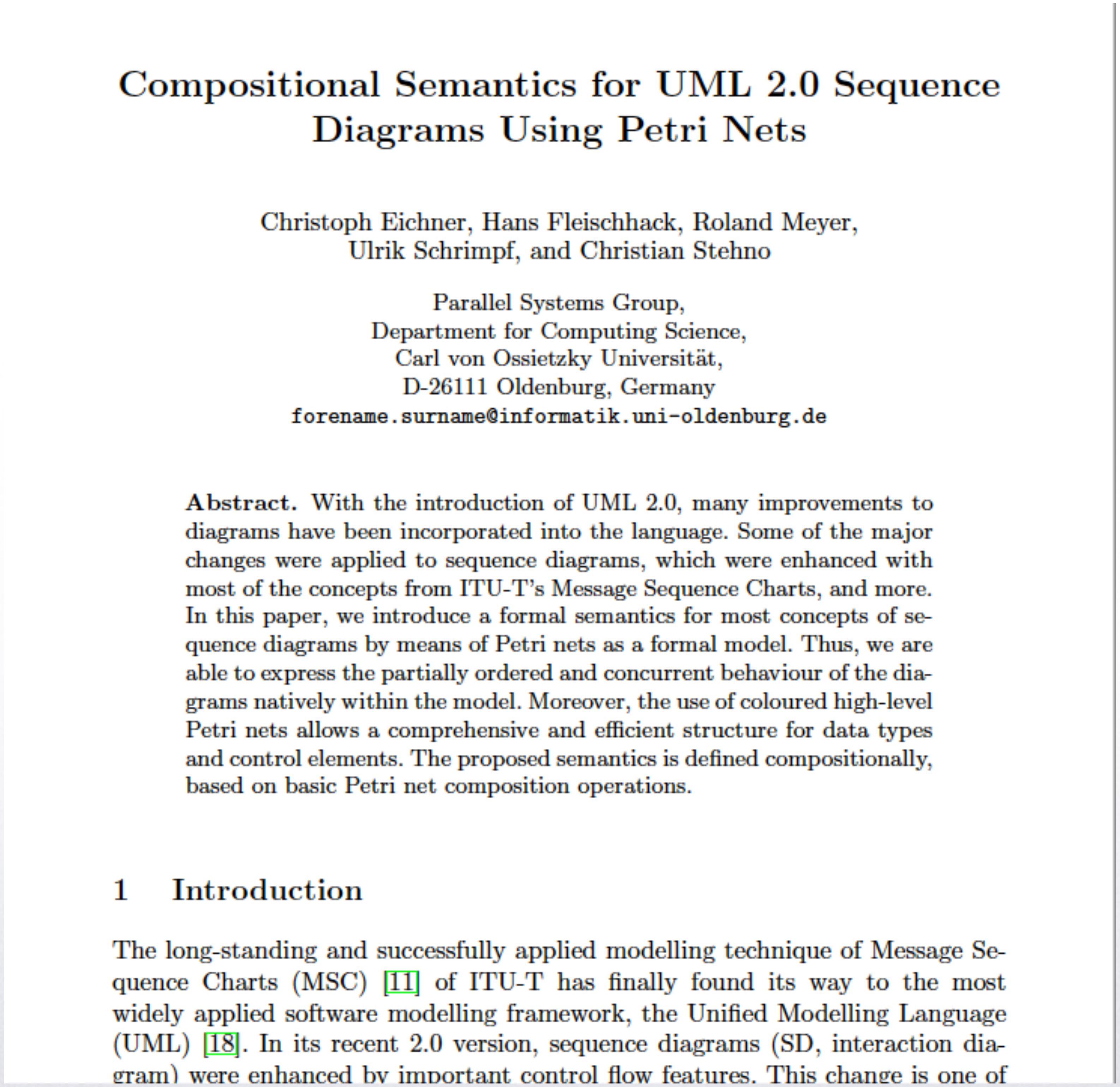
***Abstract*—With the growing use of UML diagrams for software design description and the importance of nonfunctional requirements evaluation at software development process, transforming these diagrams to executable models is considered to be significant. Nonfunctional requirements can not be evaluated directly by UML diagrams. Software designers are not usually familiar with non-functional requirement analysis and are not able to analyze such requirements easily. Therefore the designer should produce an executable model from software design description to be ready for analysis. usecase and sequence diagrams are the most important UML diagrams for software design description. In this paper, we propose new algorithms that enable a designer to transform usecase and sequence diagrams into executable models based on Petri nets and then we show how to use this Petri net models for simulation. Finally, to represent the usage of our proposed algorithms, we consider a case study as an example.**

Keywords-usecase diagram, sequence diagram, executable model, petri net, software design, nonfunctional requirement evaluation

I. Introduction

Nowadays, one of the most noticeable tasks of a designer

specific nonfunctional quality attributes and specific domains. When we apply them to other quality attributes such as reliability, we should redesign the queuing model to a new one. Elkoutbi et al. have transformed a simple usecase structure to color Petri nets [5] and kamandi et al. has transformed usecase to Object Stochastic Activity Network (OSAN) [6]. However, there have been few researches about transformation of usecase diagram to Petri net. Different approaches have been used for the transformation of sequence diagram to Petri nets. In proposed approach by Bernardi et al. all structures of sequence diagram have transformed to Generalized Stochastic Petri Nets [2]. Ourdani et al. have transformed the simplest structures in sequence diagram to colored petri net [7]. The difference between two transformations is that in Bernardi et al. approach the transformation is based on mapping of messages as well as conveying them, but in Ourdani et al. approach, the transformation is based on message sender and receiver component. Some of these researches have not utilized all structures of usecase and sequence diagrams for transformation. On the other hand, in an executable model based on Petri nets, when we add a quality attribute to be measured, we just attach the values denoting the attribute to tokens and adopt expressions for calculating the quality attribute value from the newly attached values on the tokens



Compositional Semantics for UML 2.0 Sequence Diagrams Using Petri Nets

Christoph Eichner, Hans Fleischhack, Roland Meyer,
Ulrik Schrimpf, and Christian Stehno

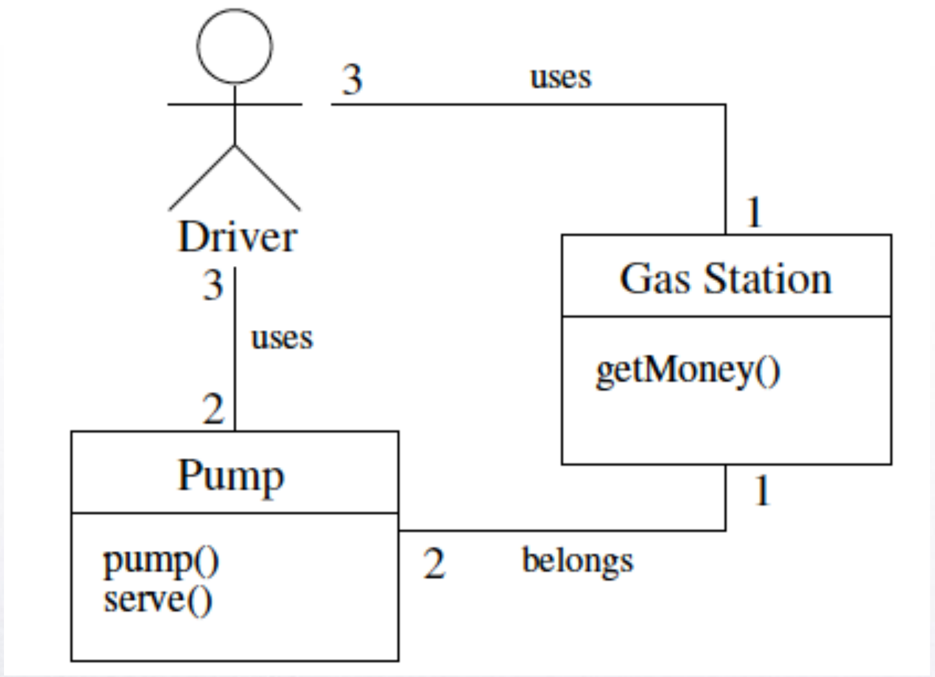
Parallel Systems Group,
Department for Computing Science,
Carl von Ossietzky Universität,
D-26111 Oldenburg, Germany
forename.surname@informatik.uni-oldenburg.de

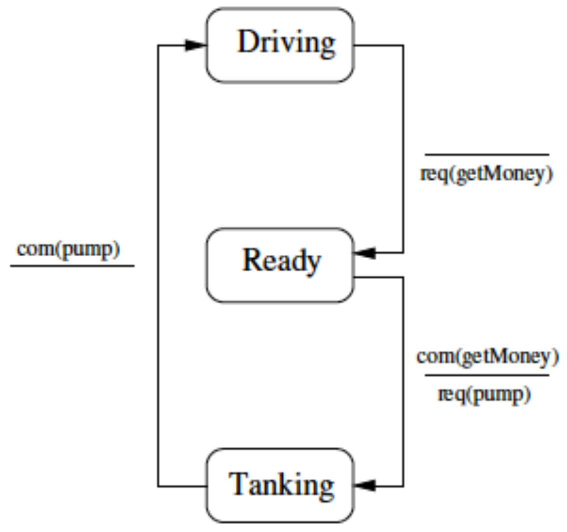
Abstract. With the introduction of UML 2.0, many improvements to diagrams have been incorporated into the language. Some of the major changes were applied to sequence diagrams, which were enhanced with most of the concepts from ITU-T's Message Sequence Charts, and more. In this paper, we introduce a formal semantics for most concepts of sequence diagrams by means of Petri nets as a formal model. Thus, we are able to express the partially ordered and concurrent behaviour of the diagrams natively within the model. Moreover, the use of coloured high-level Petri nets allows a comprehensive and efficient structure for data types and control elements. The proposed semantics is defined compositionally, based on basic Petri net composition operations.

1 Introduction

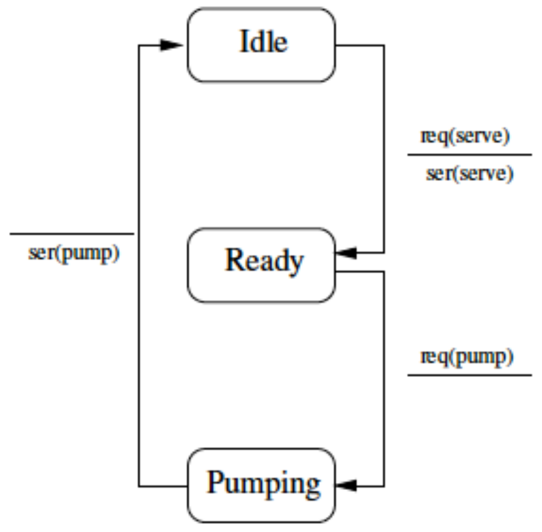
The long-standing and successfully applied modelling technique of Message Sequence Charts (MSC) [11] of ITU-T has finally found its way to the most widely applied software modelling framework, the Unified Modelling Language (UML) [18]. In its recent 2.0 version, sequence diagrams (SD, interaction diagram) were enhanced by important control flow features. This change is one of

Em um artigo de 2001, Luciano Baresi e Mauro Pezzé propuseram a síntese de uma rede de Petri a partir de diagramas de classe e estado

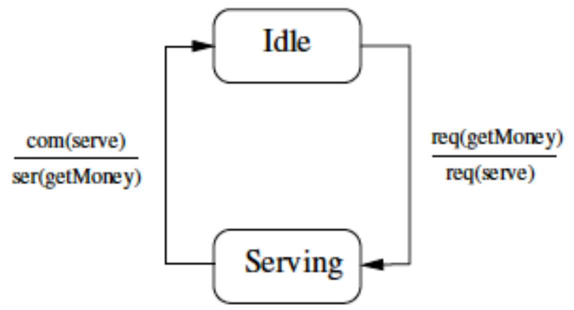




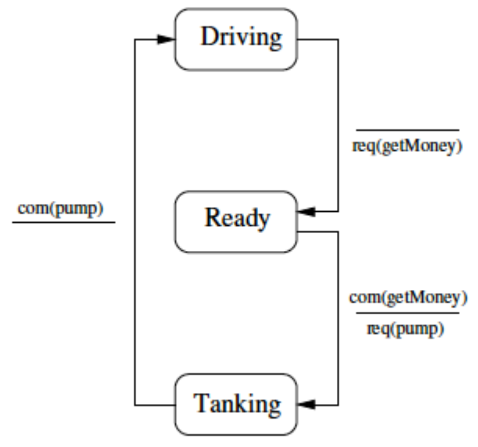
(a) *class Driver*



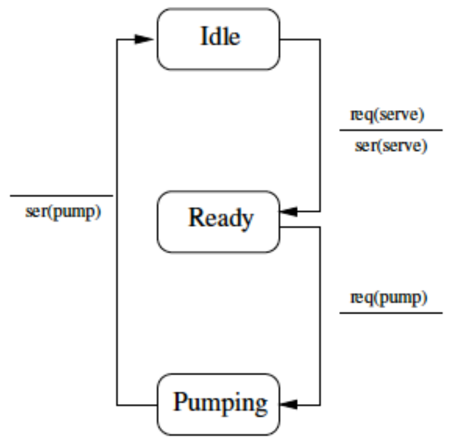
(b) *class Pump*



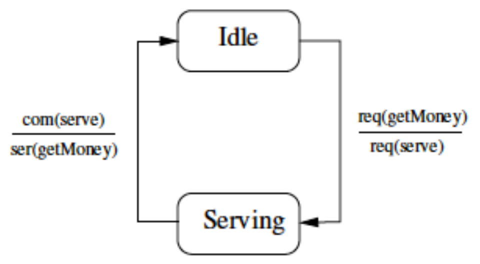
(c) *class Gas Station*



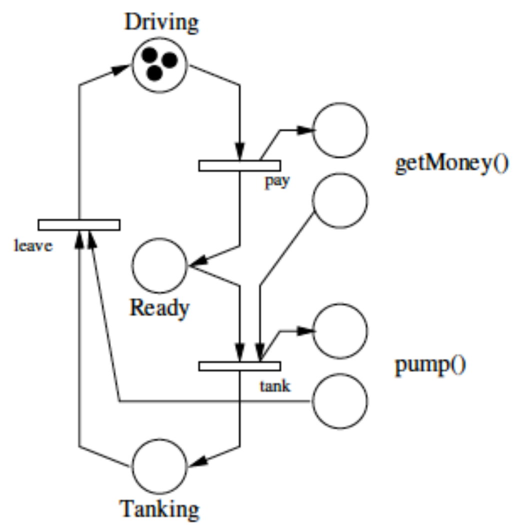
(a) class Driver



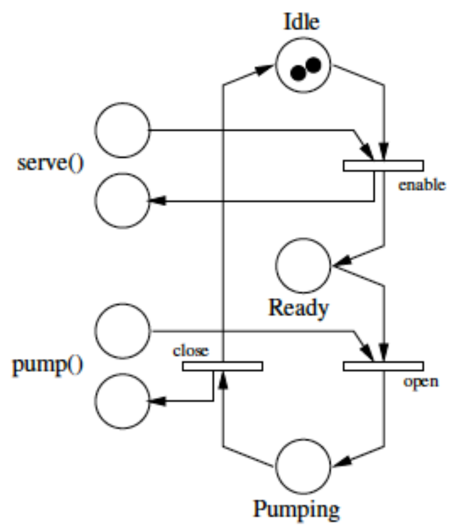
(b) class Pump



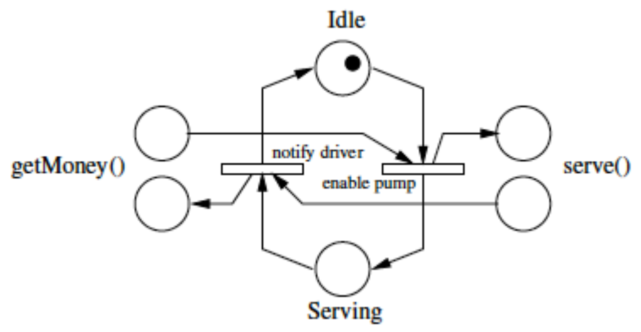
(c) class Gas Station



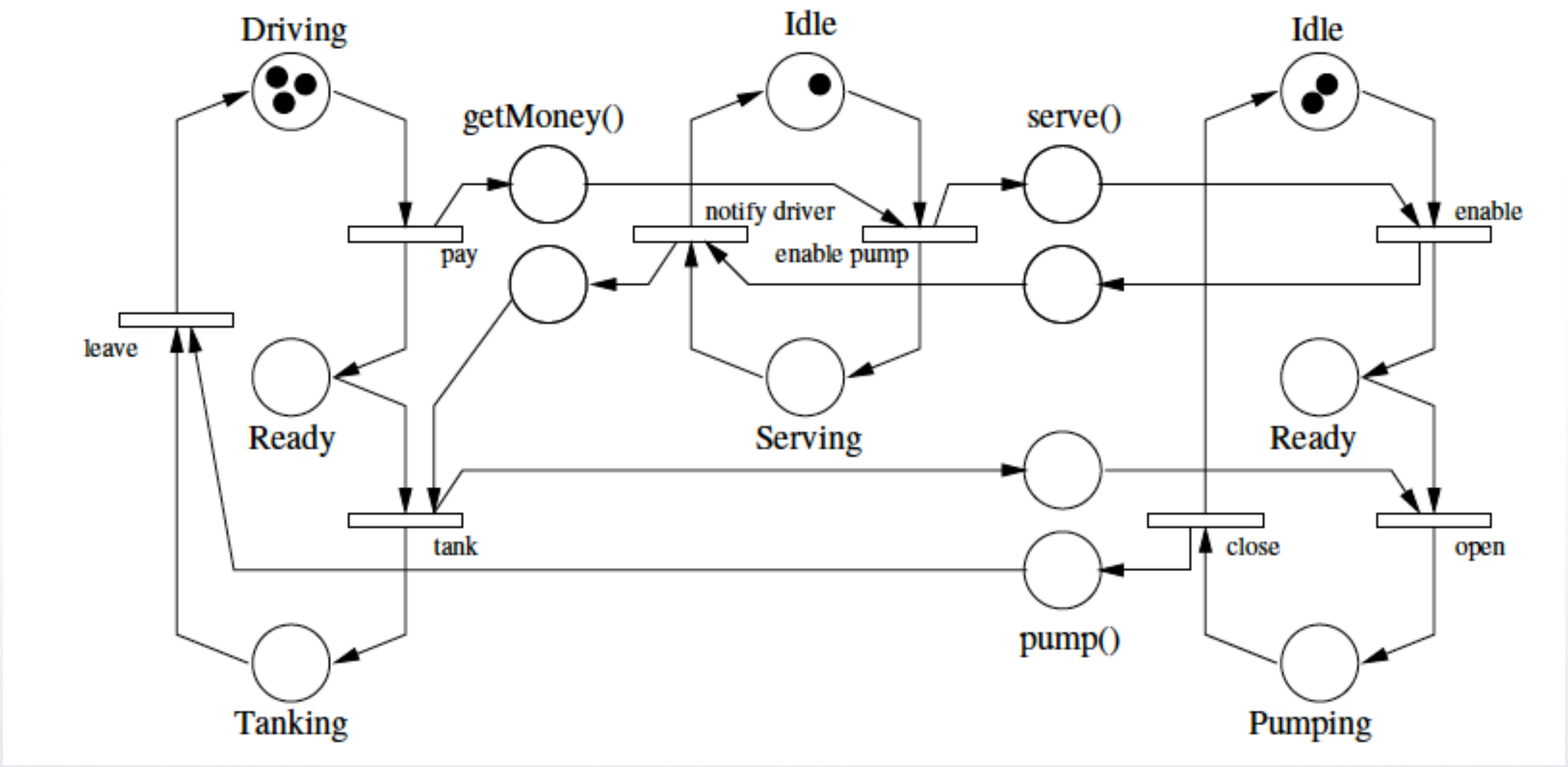
(a) class Driver

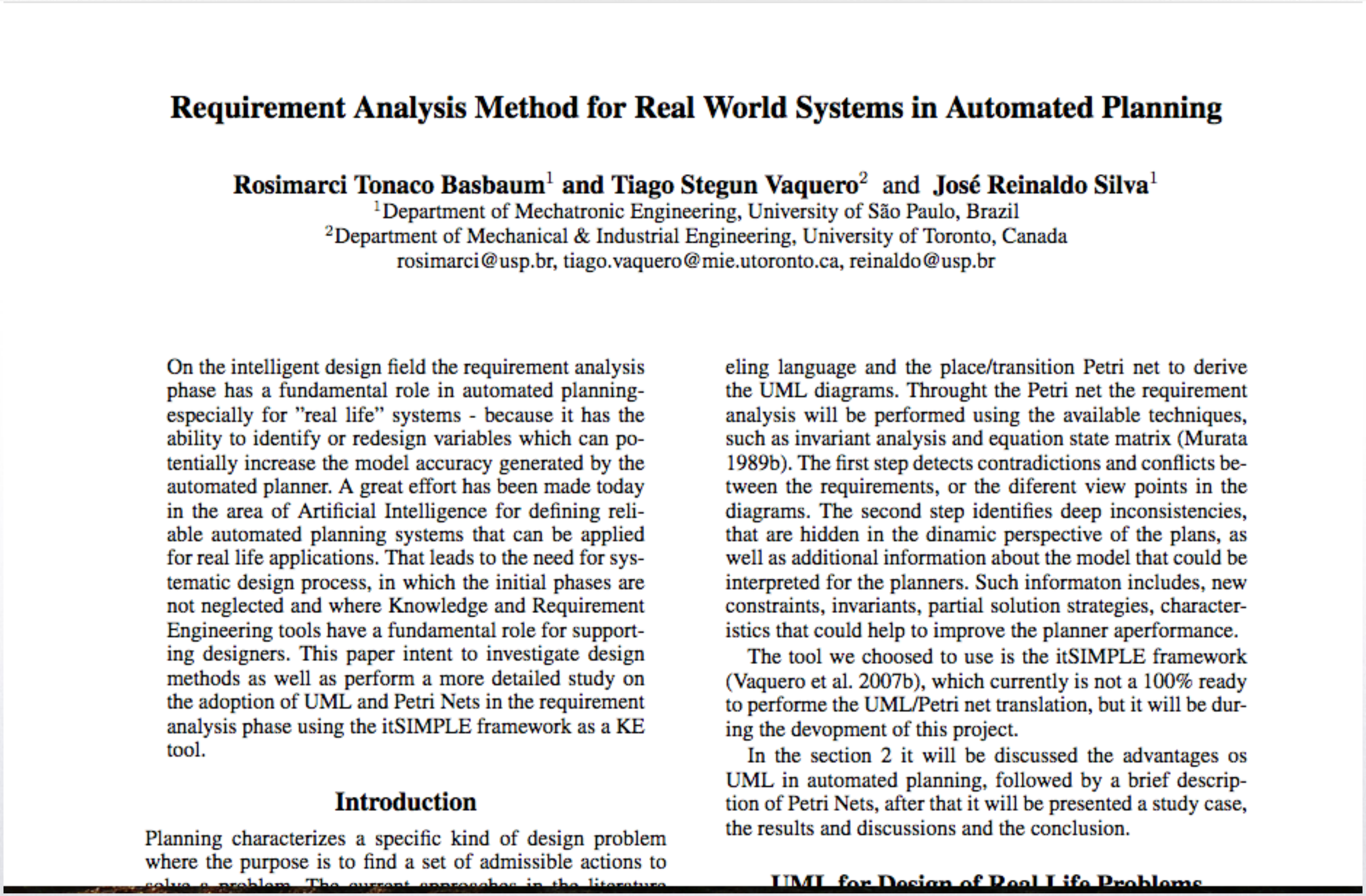


(b) class Pump



(c) class Gas Station


Driving
getMoney()
Idle
serve()
Idle
notify driver
enable
pay
enable pump
leave
Ready
Serving
Ready
tank
close
open
pump()
Tanking
Pumping



Requirement Analysis Method for Real World Systems in Automated Planning

Rosimarci Tonaco Basbaum[1] **and Tiago Stegun Vaquero**[2] and **José Reinaldo Silva**[1]
[1]Department of Mechatronic Engineering, University of São Paulo, Brazil
[2]Department of Mechanical & Industrial Engineering, University of Toronto, Canada
rosimarci@usp.br, tiago.vaquero@mie.utoronto.ca, reinaldo@usp.br

On the intelligent design field the requirement analysis phase has a fundamental role in automated planning- especially for "real life" systems - because it has the ability to identify or redesign variables which can potentially increase the model accuracy generated by the automated planner. A great effort has been made today in the area of Artificial Intelligence for defining reliable automated planning systems that can be applied for real life applications. That leads to the need for systematic design process, in which the initial phases are not neglected and where Knowledge and Requirement Engineering tools have a fundamental role for supporting designers. This paper intent to investigate design methods as well as perform a more detailed study on the adoption of UML and Petri Nets in the requirement analysis phase using the itSIMPLE framework as a KE tool.

Introduction

Planning characterizes a specific kind of design problem where the purpose is to find a set of admissible actions to solve a problem. The current approaches in the literature

eling language and the place/transition Petri net to derive the UML diagrams. Throught the Petri net the requirement analysis will be performed using the available techniques, such as invariant analysis and equation state matrix (Murata 1989b). The first step detects contradictions and conflicts between the requirements, or the diferent view points in the diagrams. The second step identifies deep inconsistencies, that are hidden in the dinamic perspective of the plans, as well as additional information about the model that could be interpreted for the planners. Such informaton includes, new constraints, invariants, partial solution strategies, characteristics that could help to improve the planner aperformance.

The tool we choosed to use is the itSIMPLE framework (Vaquero et al. 2007b), which currently is not a 100% ready to performe the UML/Petri net translation, but it will be during the devopment of this project.

In the section 2 it will be discussed the advantages os UML in automated planning, followed by a brief description of Petri Nets, after that it will be presented a study case, the results and discussions and the conclusion.

UML for Design of Real Life Problems

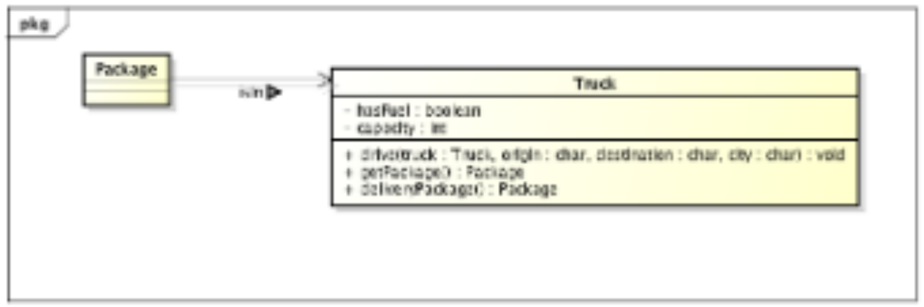

pkg
Package
isIn
Truck
- hasFuel : boolean
- capacity : int
+ drivetruck : Truck, origin : char, destination : char, city : char) : void
+ getPackage() : Package
+ deliveryPackage() : Package

Figure 2: Class Diagram designed to represent the planning problem.

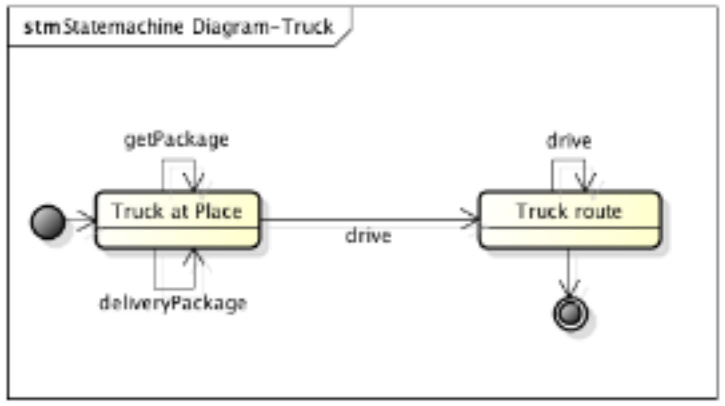


Figure 4: State Diagram designed to represent the planning problem.

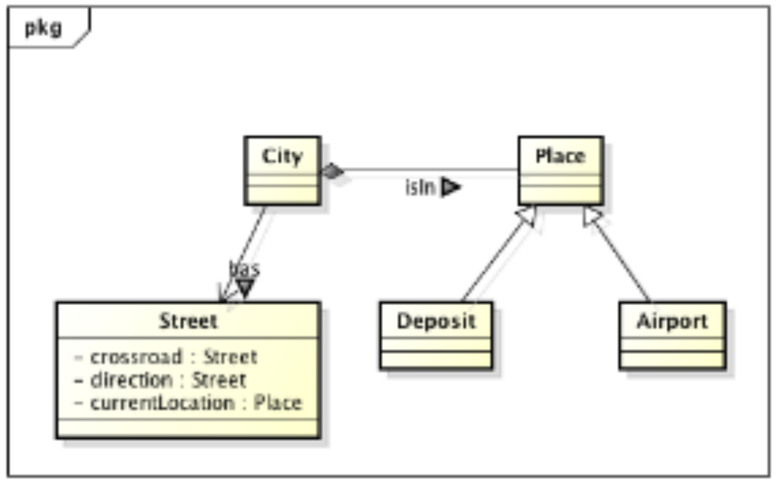


Figure 3: Class Diagram designed to represent the work domain.

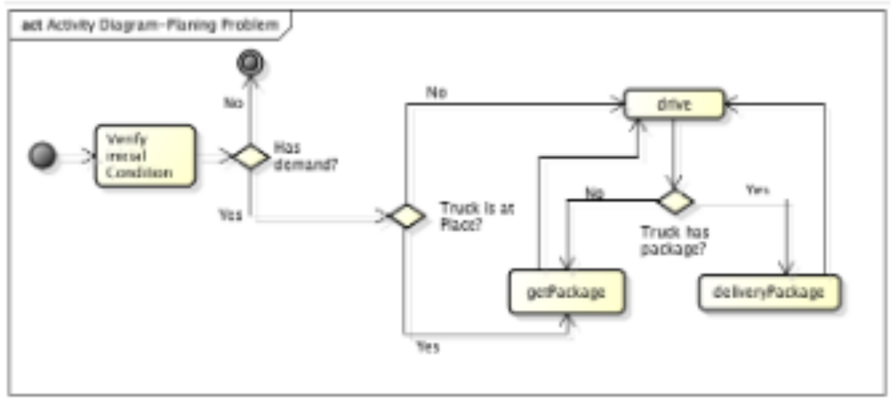


Figure 5: Activity Diagram designed to represent the planning problem.

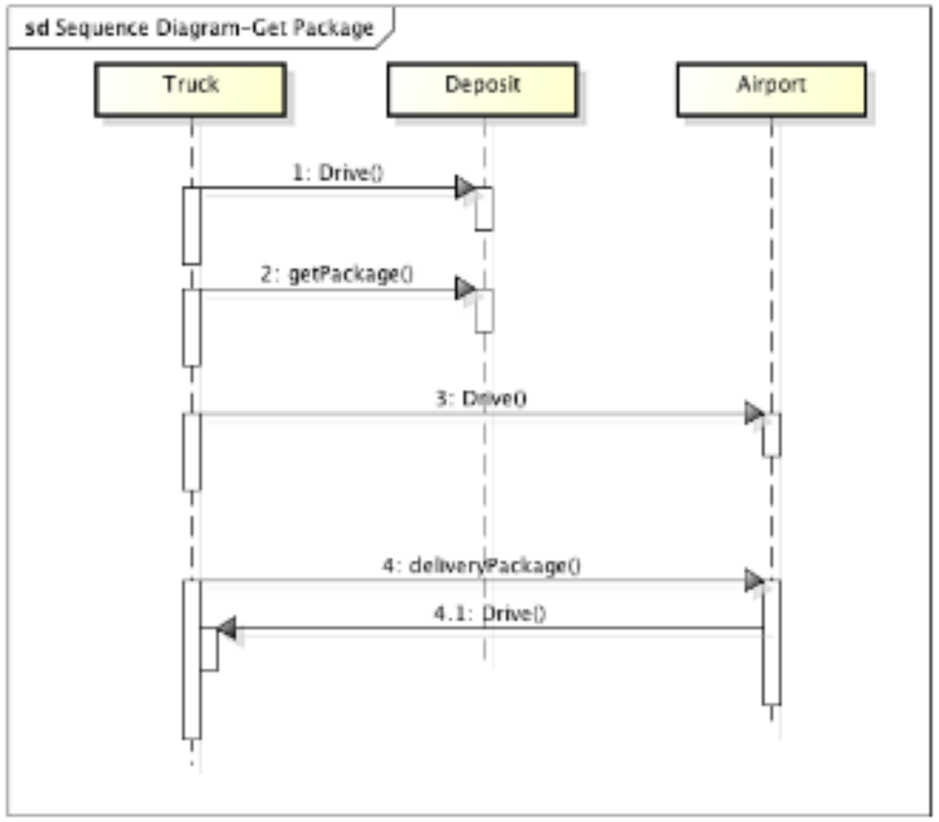
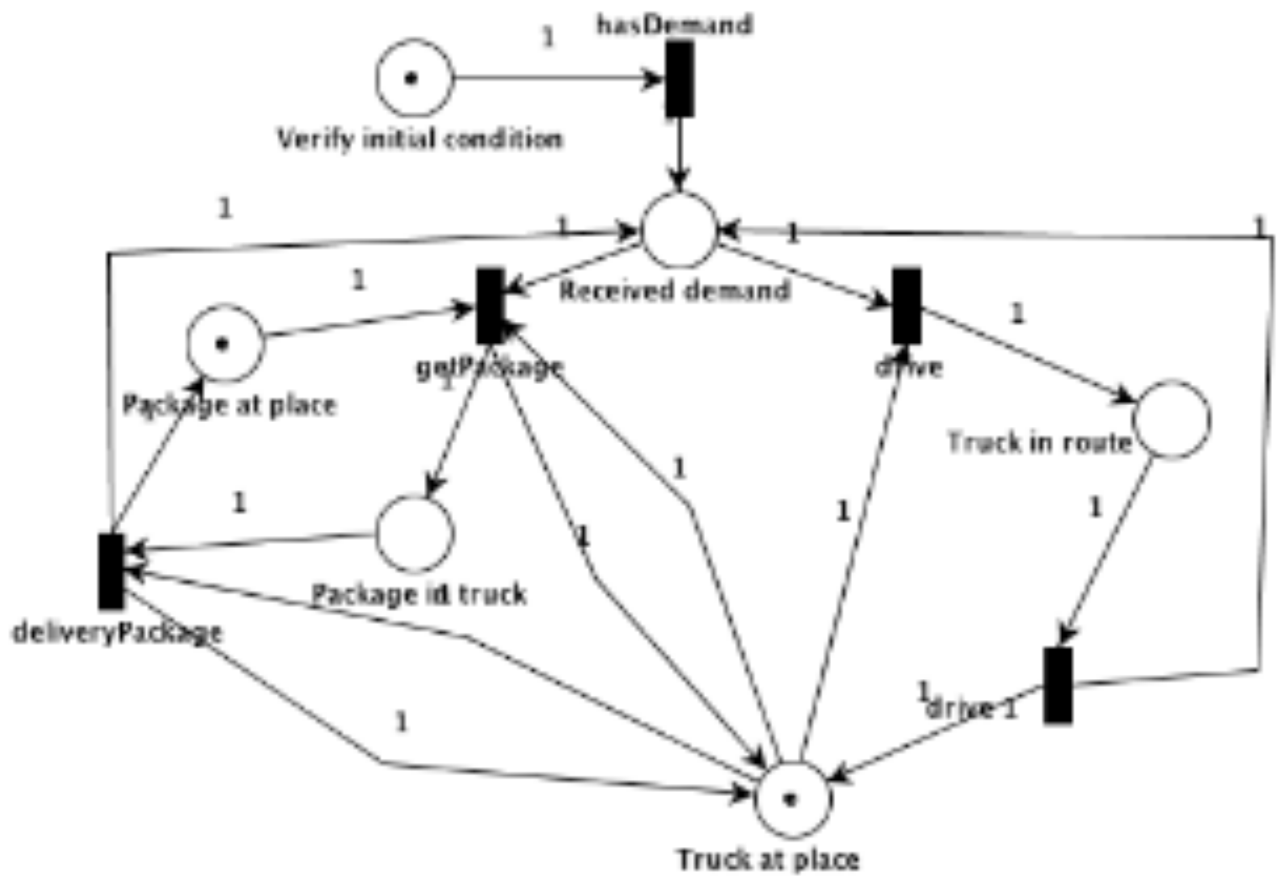


Figure 6: Sequence Diagram designed to represent the planning problem.



Fim