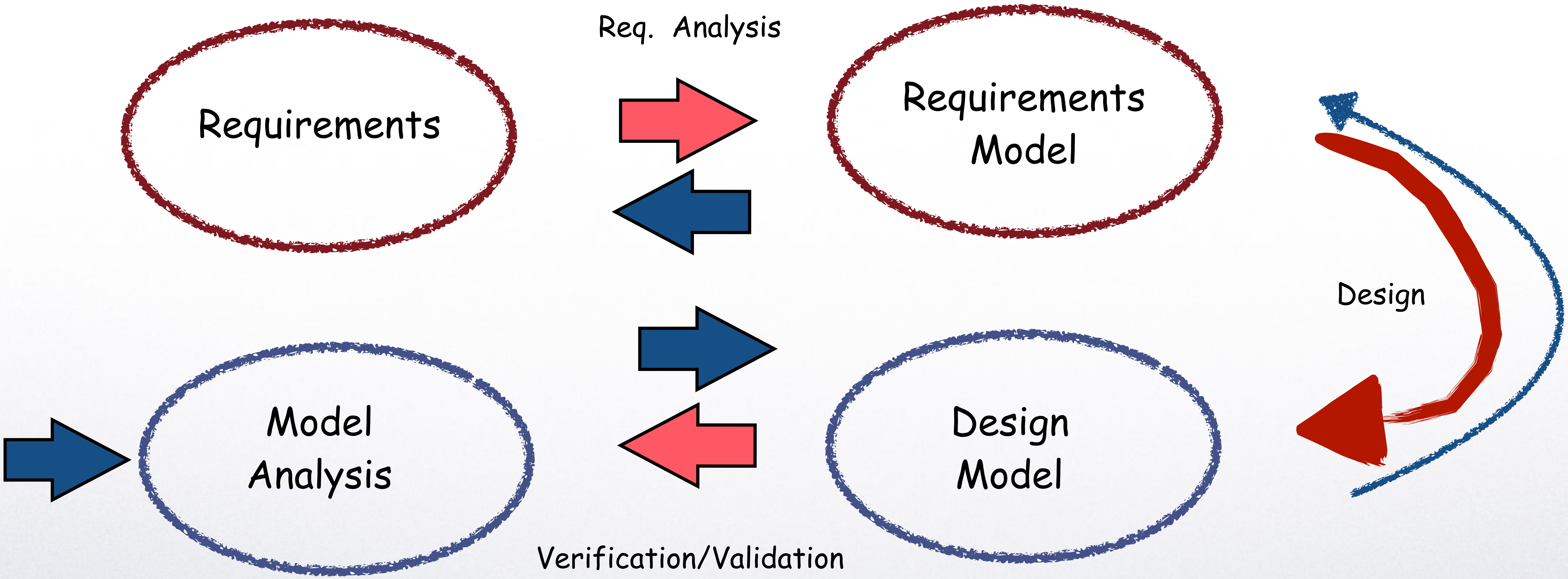
PMR 5237
Modelagem e Design de Sistemas Discretos em Redes de Petri

Aula 10 : Modelagem e análise de propriedades em RdP: invariantes

Prof. José Reinaldo Silva
reinaldo@usp.br

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).

Fairness

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.

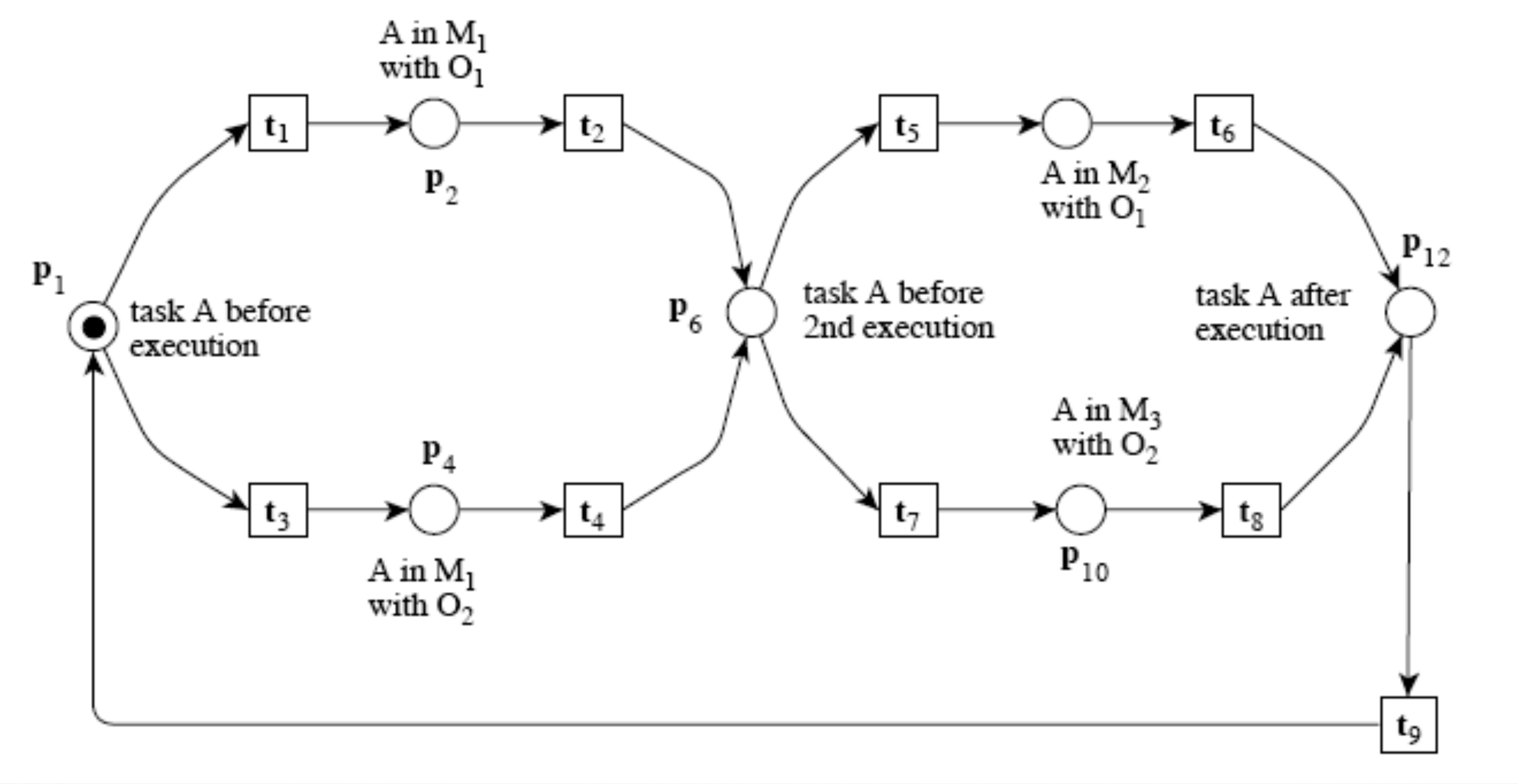
Modelagem de Sistemas Discretos

- 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.



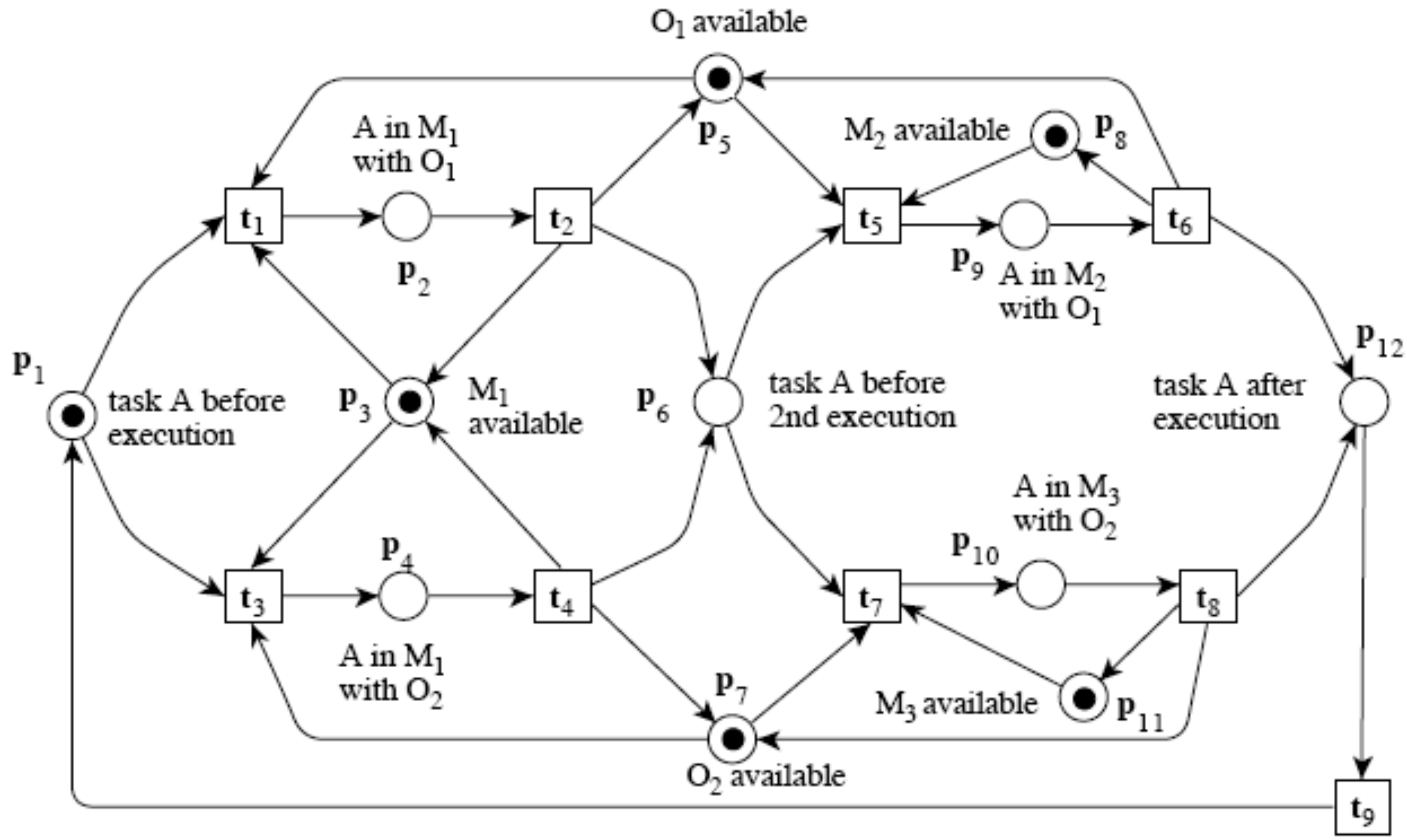
Girauld, C. and Valk, R.; Petri Nets for Systems Engineering, Springer-Verlag, 2003

Análise

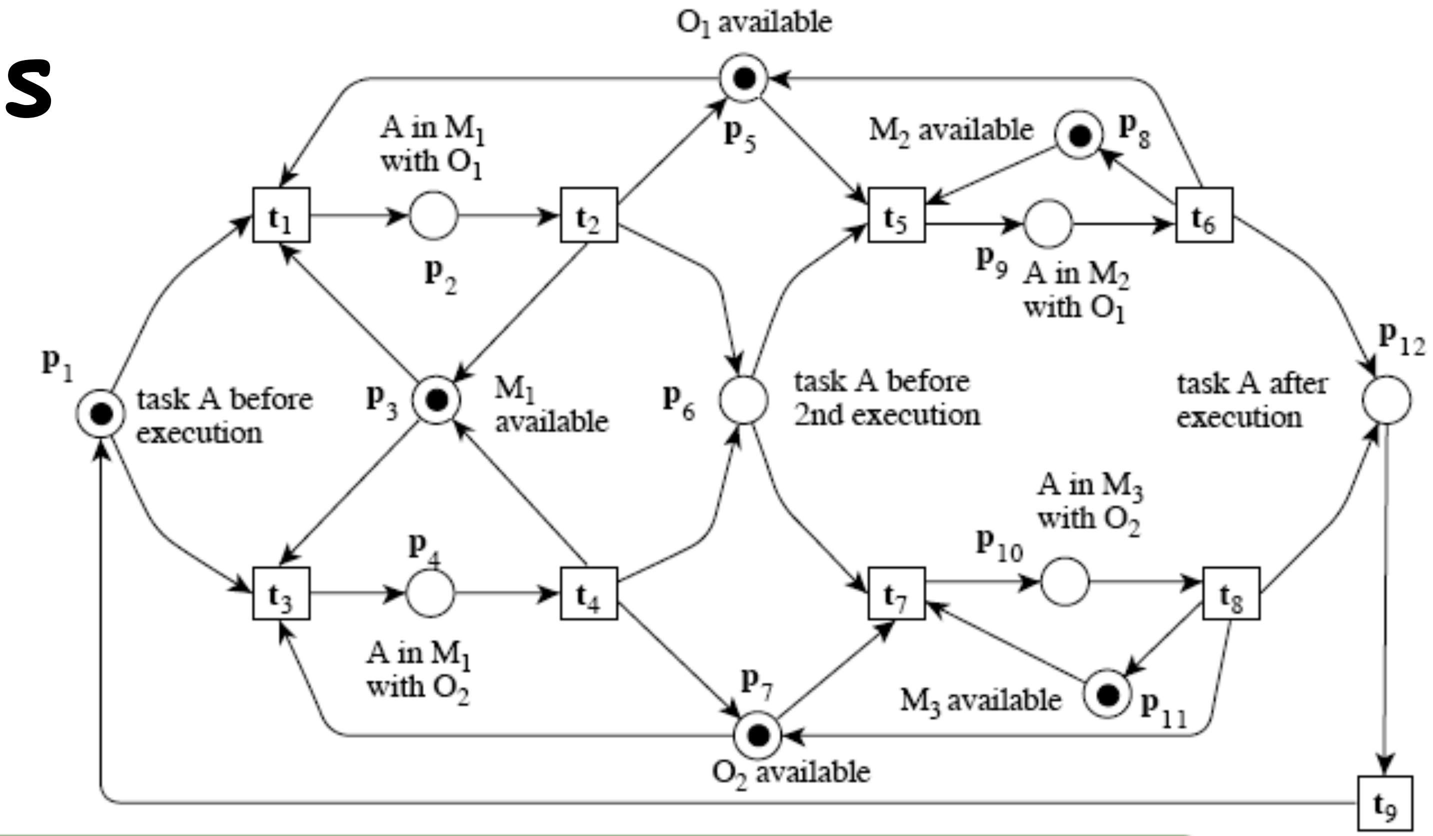
O sistema é cíclico, e permite a combinação das operações em qualquer ordem (e portanto o sistema estaria preparado para implementar qualquer receita de peça). O sistema é conservativo, de modo que cada peça seria representada por uma marca que deve estar em algum dos lugares já especificados.

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

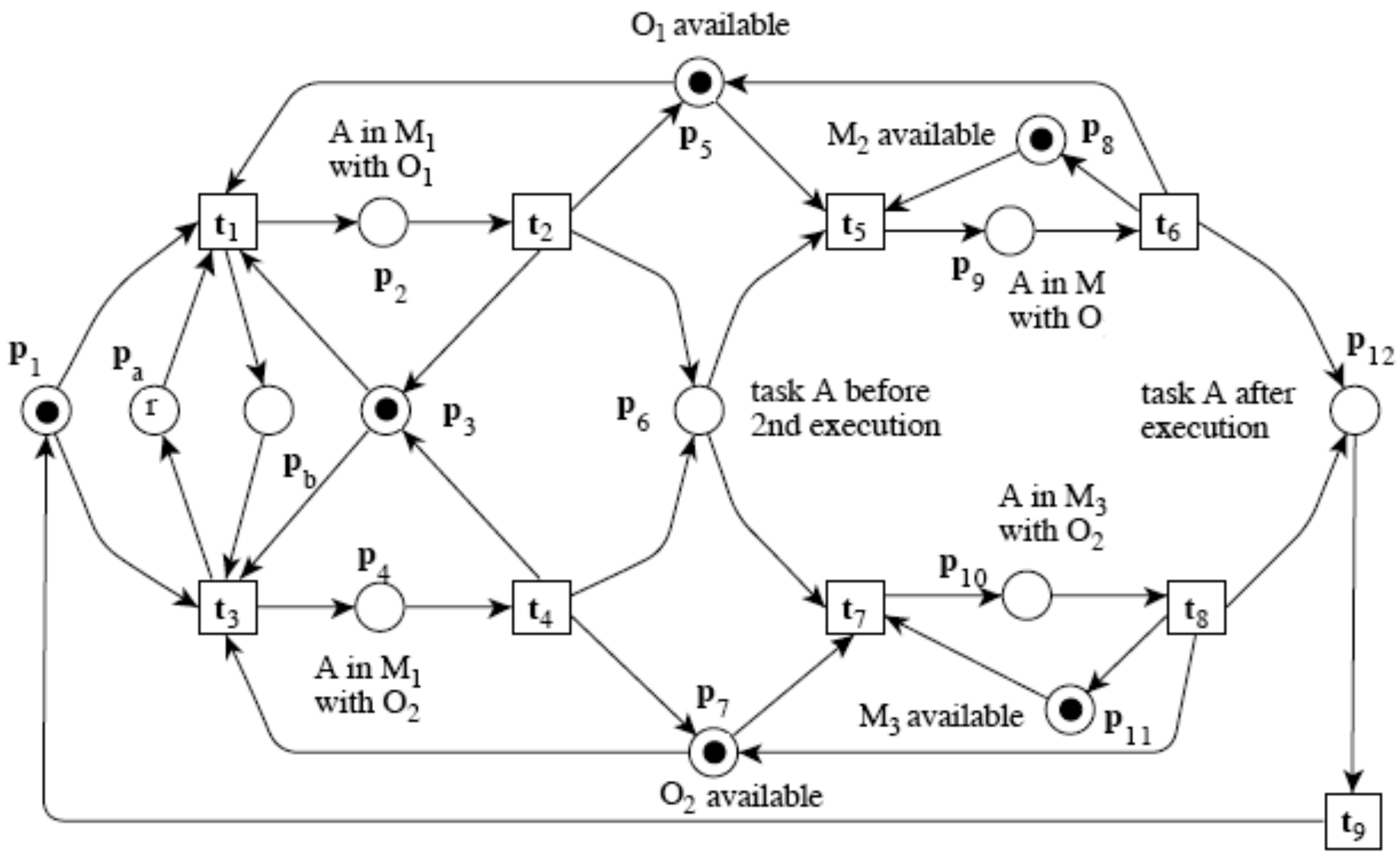
i) $m[p_2] + m[p_5] + m[p_9] = 1;$

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

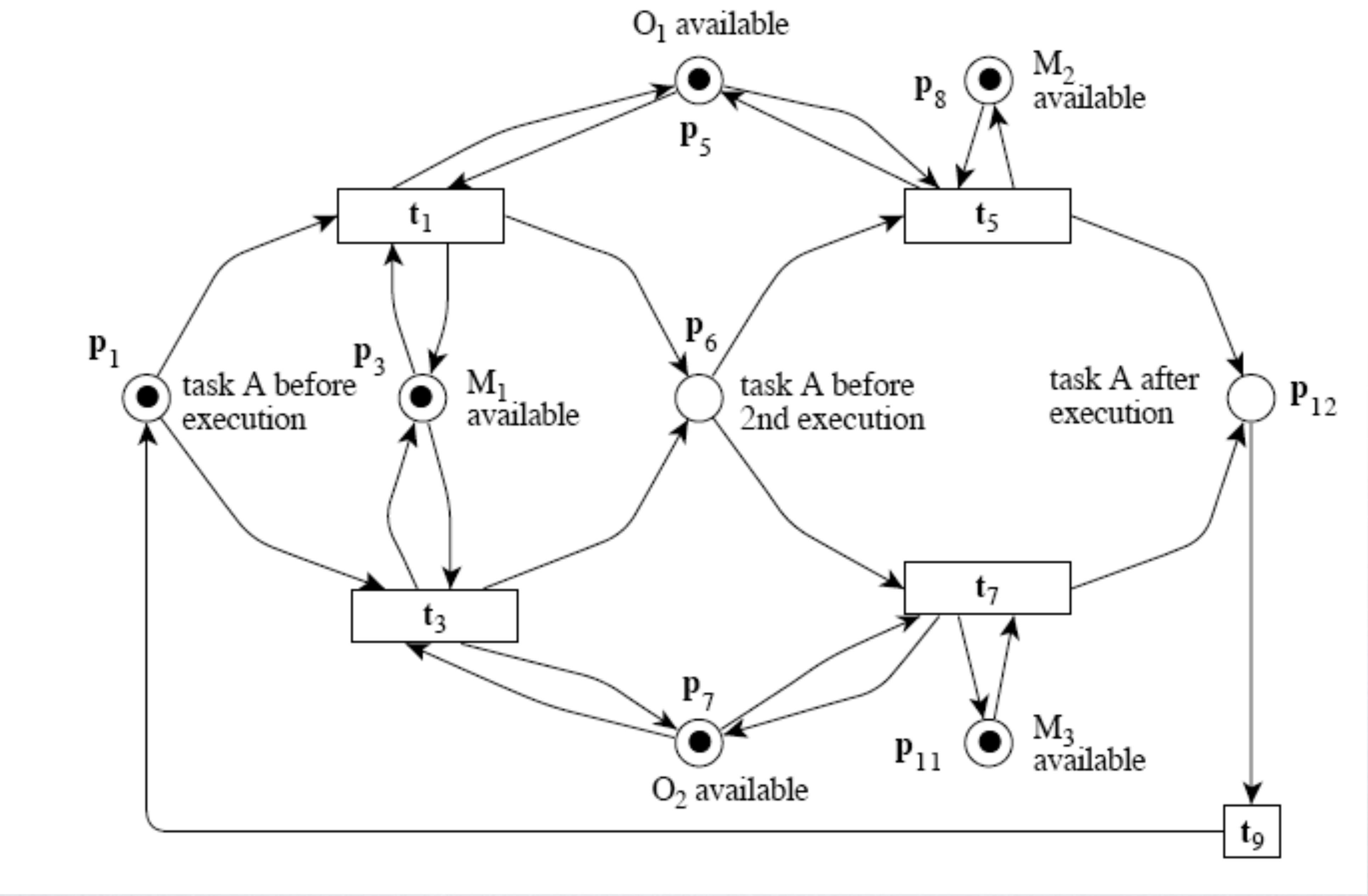
Definition 45

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

Invariant Analysis

Invariant analysis is concerned with a perception of conservation laws in the flux of changing, for instance of states. Therefore, in state-transition (or transition systems) analysis, invariants could be associated with structural properties which could also affect systems behaviour.

A *p-flow* (*t-flow*) is a vector $\mathbf{y} : P \to \mathbb{Q}$ such that $\mathbf{y} \cdot \mathbf{C} = 0$ ($\mathbf{x} : T \to \mathbb{Q}$ such that $\mathbf{C} \cdot \mathbf{x} = 0$), where $\mathbf{C}$ is the incidence matrix of the net. The set of p-flows (t-flows) is a vector space, orthogonal to the space of the rows (columns) of $\mathbf{C}$.

Therefore, the flows can be generated from a *basis* of the space. Natural (i.e. non-negative integer) p-flows (t-flows) are called *p-semiflows* (*t-semiflows*): vectors $\mathbf{y} : P \to \mathbb{N}$ such that $\mathbf{y} \cdot \mathbf{C} = 0$ ($\mathbf{x} : T \to \mathbb{N}$ such that $\mathbf{C} \cdot \mathbf{x} = 0$).

If y.C=0 (C.x=0), we are in fact looking for a basis of vectors that compose the “kernel" of C space. That means the identification of the set of places (transitions) that remains invariant (preserving marks or leading to the same state).

Given a p (t) semiflow, we will define its support set as:

$$\| y \| = p \in P \mid y(p) > 0$$

$$\| x \| = t \in T \mid x(t) > 0$$

A p-(t-)-semiflow is said to be minimal (or proper) iff it is canonical, that is, if all its non null elements are equal to 1, and it support set is unique, that is, different of any other p-(t-)semiflow.

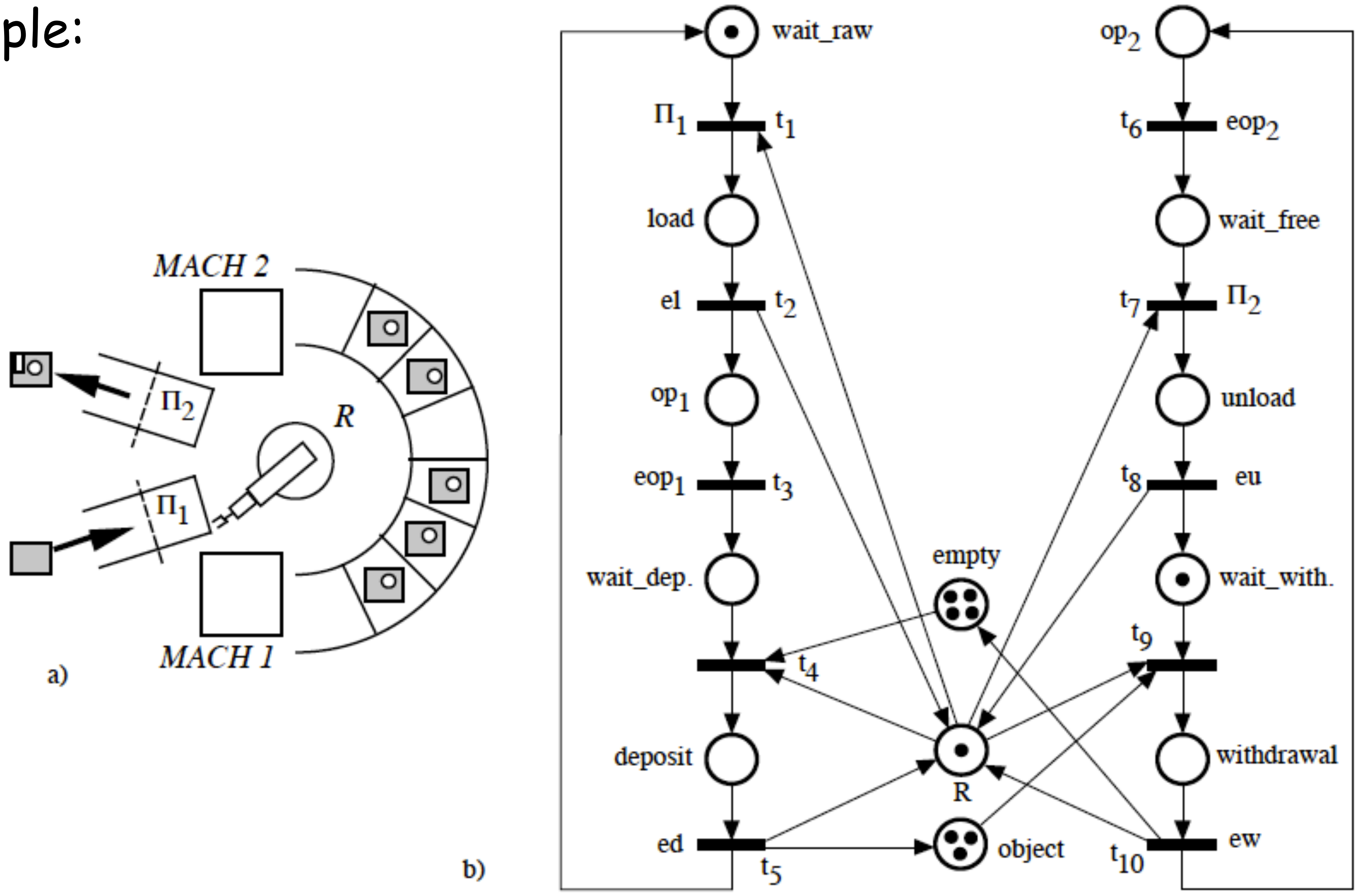
Conservation laws obtained from semiflows:

$$\mathbf{y} \in \mathbb{N}^n \wedge \mathbf{y} \cdot \mathbf{C} = 0 \Longrightarrow \forall \mathbf{m_0} \,.\, \forall \mathbf{m} \in \mathrm{RS}(\mathcal{N}, \mathbf{m_0}) \,.\, [\mathbf{y} \cdot \mathbf{m} = \mathbf{y} \cdot \mathbf{m_0}].$$

$$\mathbf{x} \in \mathbb{N}^m \wedge \mathbf{C} \cdot \mathbf{x} = 0 \Longrightarrow \exists \mathbf{m_0} \,.\, \exists \text{occurrence sequence } \sigma \,.\, [\mathbf{m_0} \xrightarrow{\sigma} \mathbf{m_0} \wedge \boldsymbol{\sigma} = \mathbf{x}]$$

where $\boldsymbol{\sigma}$ is the Parikh mapping of the occurrence sequence σ.

Example:

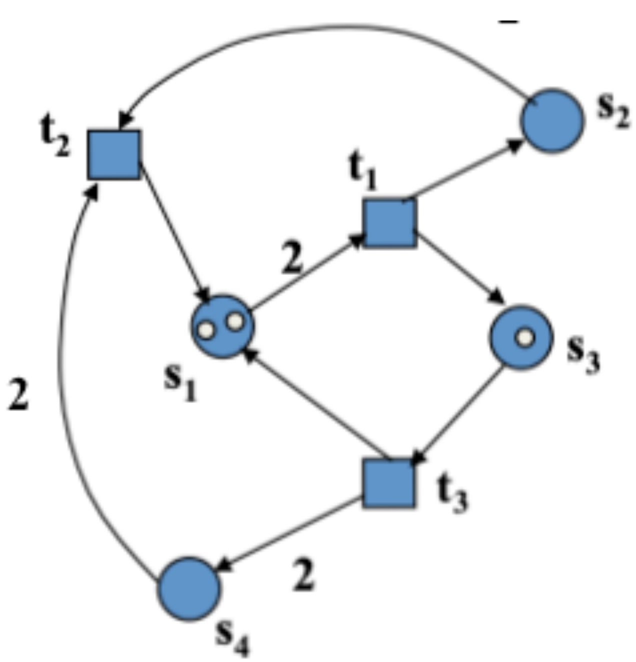

MACH 2
MACH 1
Π_2
Π_1
R
a)
b)
wait_raw
Π_1
t_1
load
el
t_2
op_1
eop_1
t_3
wait_dep.
t_4
deposit
ed
t_5
empty
R
object
op_2
t_6
eop_2
wait_free
t_7
Π_2
unload
t_8
eu
wait_with.
t_9
withdrawal
t_{10}
ew

$$\mathbf{m}[\text{wait\_raw}]+\mathbf{m}[\text{load}]+\mathbf{m}[\text{op}_1]+\mathbf{m}[\text{wait\_dep.}]+\mathbf{m}[\text{deposit}]=1$$
$$\mathbf{m}[\text{op}_2]+\mathbf{m}[\text{wait\_free}]+\mathbf{m}[\text{unload}]+\mathbf{m}[\text{wait\_with.}]+\mathbf{m}[\text{withdrawal}]=1$$
$$\mathbf{m}[\text{empty}]+\mathbf{m}[\text{deposit}]+\mathbf{m}[\text{object}]+\mathbf{m}[\text{withdrawal}]=7$$
$$\mathbf{m}[\text{R}]+\mathbf{m}[\text{load}]+\mathbf{m}[\text{unload}]+\mathbf{m}[\text{deposit}]+\mathbf{m}[\text{withdrawal}]=1$$

Bounds: $\forall p_i \in P\backslash\{\text{empty}, \text{object}\}\,.\,(\mathbf{m}[p_i] \leq 1 \quad \wedge \quad \mathbf{m}[\text{empty}] \leq 7 \quad \wedge \quad \mathbf{m}[\text{object}] \leq 7)$.

The places in each of the following sets are in marking mutual exclusion:
a) {wait_raw, load, op_1, wait_dep., deposit}
b) {op_2, wait_free, unload, wait_with., withdrawal}
c) {R, load, unload, deposit, withdrawal}

No exemplo utilizado até aqui ...



$$\begin{pmatrix}2\\0\\1\\0\end{pmatrix} \underset{t_1}{\Rightarrow} \begin{pmatrix}0\\1\\2\\0\end{pmatrix} \underset{t_3}{\Rightarrow} \begin{pmatrix}1\\1\\1\\2\end{pmatrix}$$

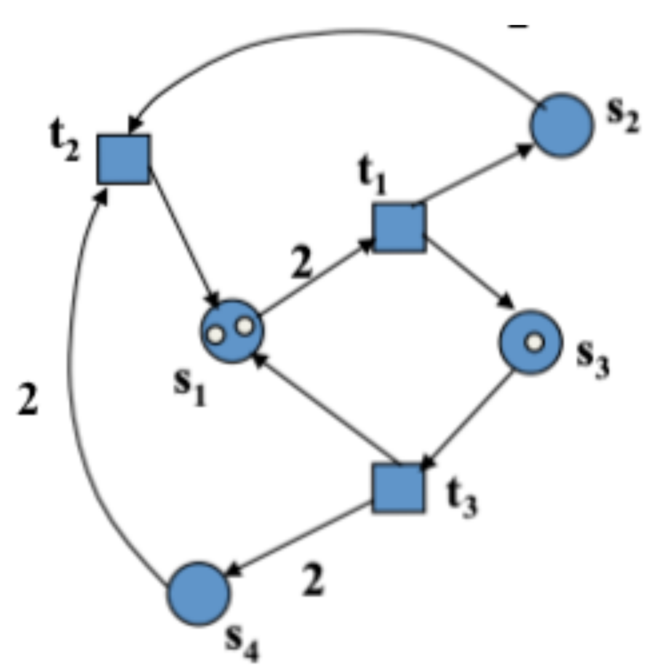
$$B_f \triangle M = B_f \left[\begin{pmatrix}2\\0\\1\\0\end{pmatrix} - \begin{pmatrix}0\\1\\2\\0\end{pmatrix} \right] = \begin{pmatrix}1 & 0 & 2 & 1/2\\0 & 1 & -1 & -1/2\end{pmatrix} \left[\begin{pmatrix}2\\-1\\-1\\0\end{pmatrix} \right] = \begin{pmatrix}0\\0\end{pmatrix}$$

Voltando à equação de estado podemos agora investigar a dinâmica da rede, e os ciclos,

$$\mathbf{A}^T \sum_{j=0}^{i} \sigma_j = \mathbf{A}^T \overline{\sigma} = \mathbf{M}_{i+1} - \mathbf{M}_0 = \Delta\mathbf{M}$$

Podemos agora selecionar os ciclos, isto é, estados e sequências de disparo tais que ΔM=0.

À soma dos vetores de habilitação que caracterizam este processo chamamos de T-invariante.



$$\begin{pmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & -2 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{aligned} -2x + y + z &= 0 \\ x - y &= 0 \\ -2y + 2z &= 0 \end{aligned}$$

$$\therefore \quad x = y = z$$

History

The formal study of algebraic (linear) invariants began in 1840 with the seminal paper of George Boole, "Exposition of a General Theory of Linear Transformations". The idea was to identify functions (or vectors) that do not depend on the vetorial basis of the linear transformation.

Invariant Calculus was developed since the middle of the lats century by several authors, but Manuel Silva authored the principal papers in Petri nets invariant analysis. The problem consists in solving an homogeneous system of equations

$$y\,.\,C = 0$$

$$C\,.\,x = 0$$

Methods to solve these equations:

Mathematic linear programming (Simplex);

Gaus-Jordan method

Algoritmo para calcular invariantes de lugar

Tese de doutorado: Modelagem e Análise de Requisitos de Sistemas Automatizados
Usando UML e Redes de Petri, Arianna Z. Olivera Salmon

Passo 1: Para obter os invariantes de lugar é preciso em primeiro lugar calcular a transposta da matriz de incidência(AT).

Passo 2: Em seguida deve-se aplicar o método de Gauss- Jordan para transformar a matriz de incidência em uma matriz diagonal. Se o posto da matriz de incidência R(AT) for menor que o numero de incógnitas (numero de lugares da rede), então ir ao passo 3. Se o posto da matriz de incidência for igual ao numero de incógnitas, então terminar, o sistema não tem invariantes de lugar.

Passo 3: Obter a solução básica do sistema representado pela matriz de incidência.

Algoritmo para calcular invariantes de transição

Tese de doutorado: Modelagem e Análise de Requisitos de Sistemas Automatizados
Usando UML e Redes de Petri, Arianna Z. Olivera Salmon

Passo 1: Calcular a matriz de incidência do sistema modelado (A).

Passo 2: Aplicar o método de Gauss- Jordan para transformar a matriz de incidência em uma matriz diagonal. Se o posto da matriz de incidência R(A) for menor que o numero de incógnitas (numero de lugares da rede), então ir ao passo 3. Se e o posto da matriz de incidência for igual ao numero de incógnitas, então terminar, o sistema não tem invariantes de transição.

Passo 3: Obter a solução básica do sistema representado pela matriz de incidência.

Propriedades estruturais	Condições necessárias e suficientes
Limitação Estrutural	$\exists x > 0,\ A^T \cdot x \geq 0$
Conservatividade	$\exists x > 0,\ A^T \cdot x = 0$
Repetitividade	$\exists y > 0,\ A \cdot y \geq 0$
Consistência	$\exists y > 0,\ A \cdot y = 0$

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

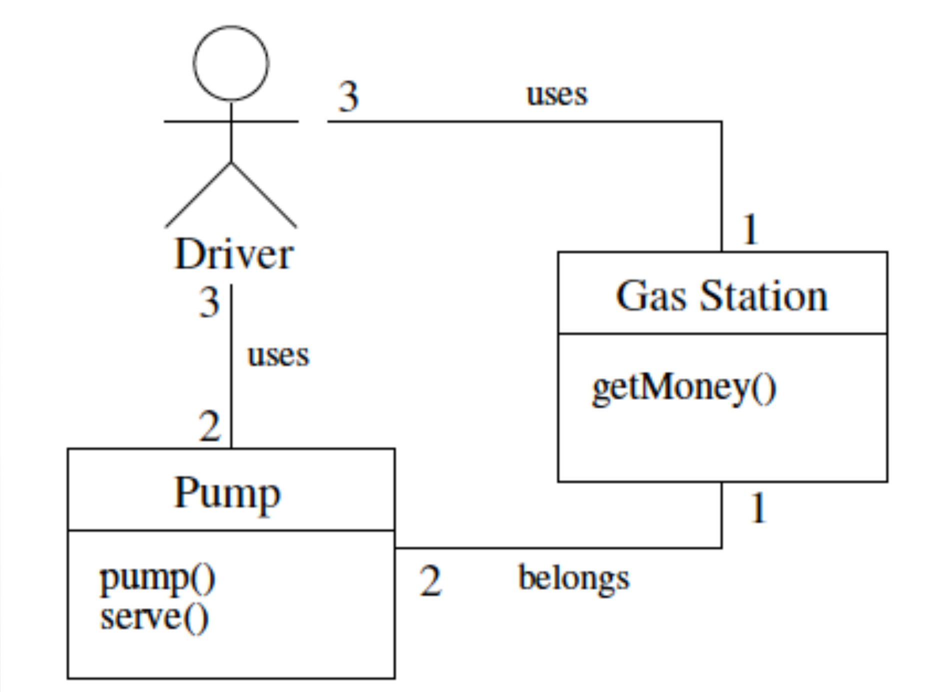
Parallel Systems Group,
Department for Computing Science,
Carl von Ossietzky Universität,
D-26111 Oldenburg, Germany
forename.surname@informatik.uni-oldenburg.de

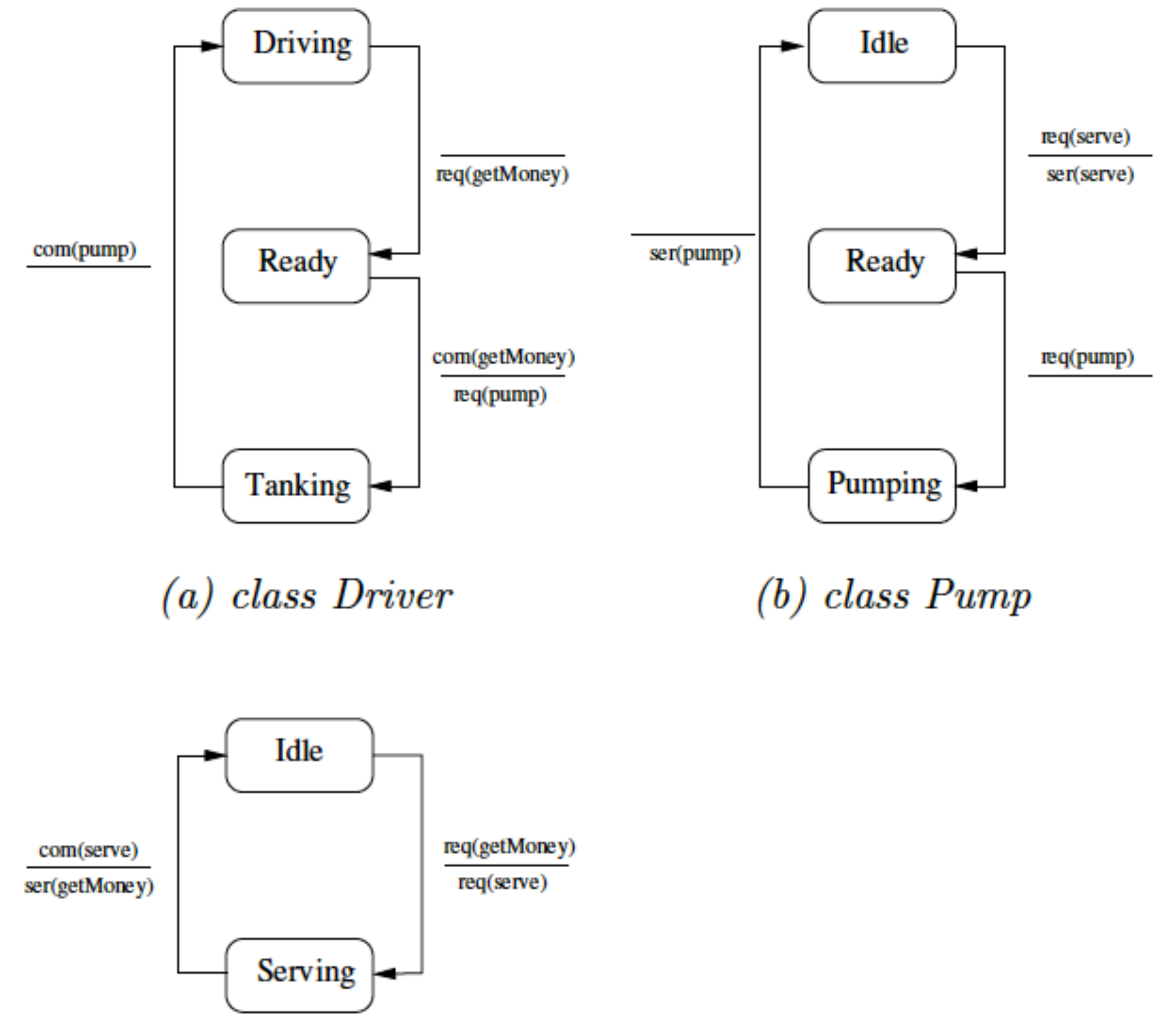
Abstract. With the introduction of UML 2.0, many improvements to diagrams have been incorporated into the language. Some of the major changes were applied to sequence diagrams, which were enhanced with most of the concepts from ITU-T's Message Sequence Charts, and more. In this paper, we introduce a 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

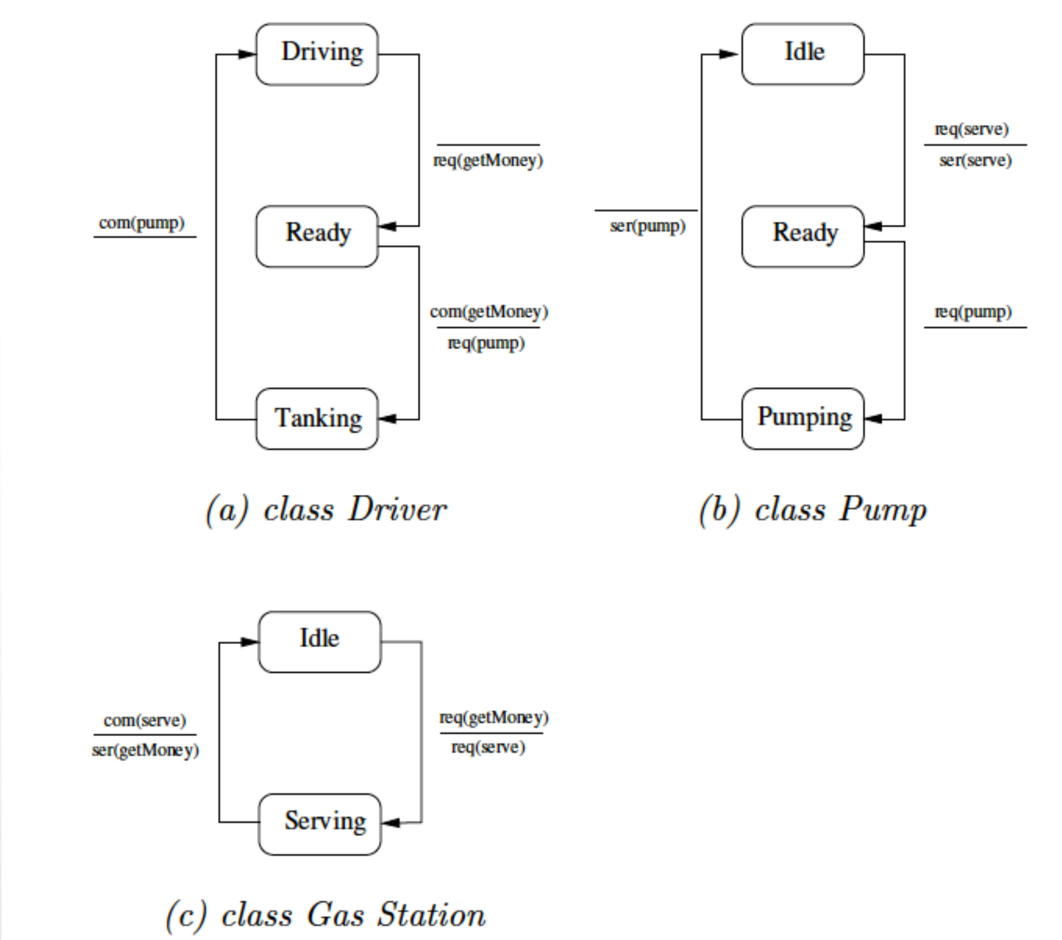


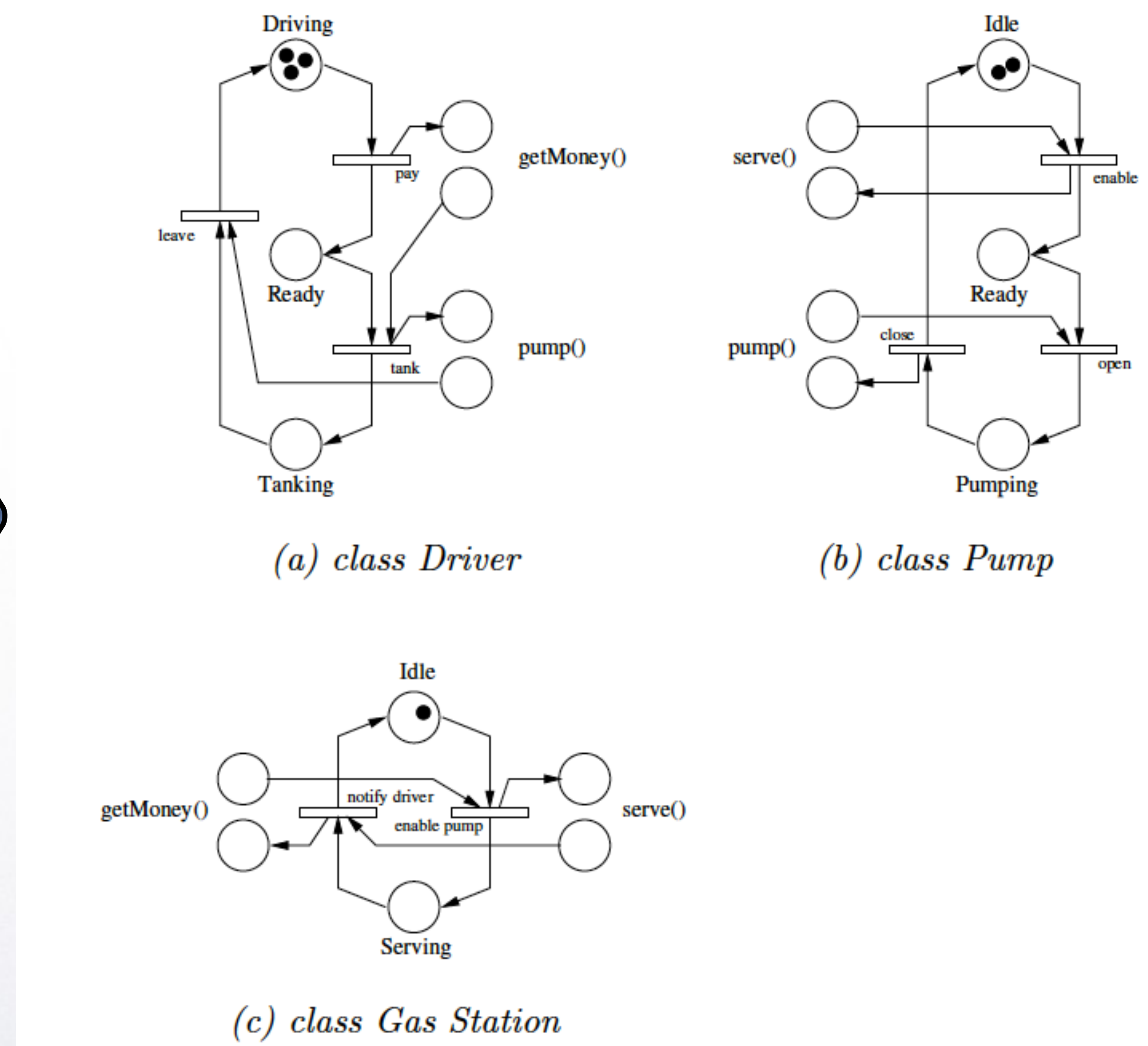


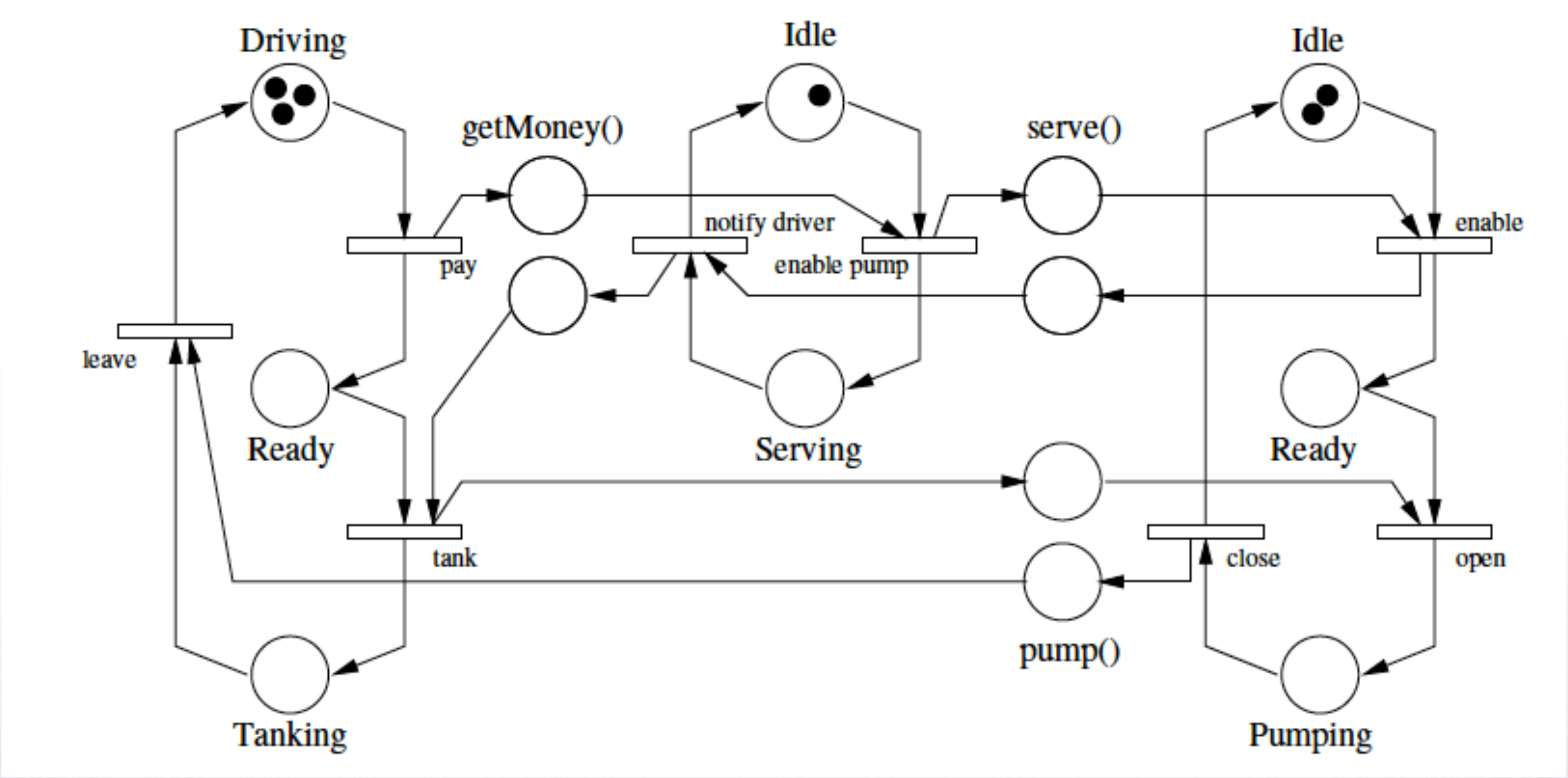
(a) class Driver

(b) class Pump

(c) class Gas Station


Driving
Ready
Tanking
com(pump)
req(getMoney)
com(getMoney)
req(pump)
(a) class Driver
Idle
Ready
Pumping
ser(pump)
req(serve)
ser(serve)
req(pump)
(b) class Pump
Idle
Serving
com(serve)
ser(getMoney)
req(getMoney)
req(serve)
(c) class Gas Station


Driving
getMoney()
pay
leave
Ready
pump()
tank
Tanking
(a) class Driver
Idle
serve()
enable
Ready
pump()
close
open
Pumping
(b) class Pump
Idle
getMoney()
notify driver
enable pump
serve()
Serving
(c) class Gas Station


Driving
Idle
Idle
getMoney()
serve()
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

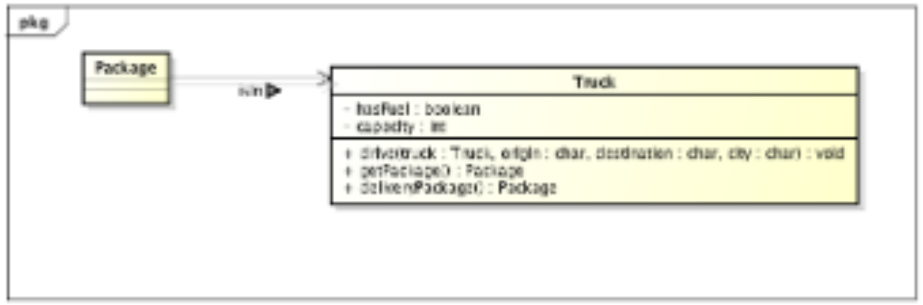


Figure 2: Class Diagram designed to represent the planning problem.

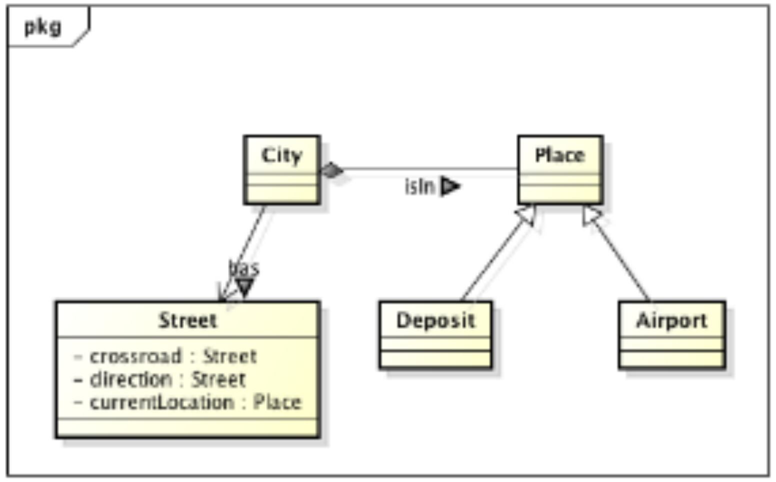


Figure 3: Class Diagram designed to represent the work domain.

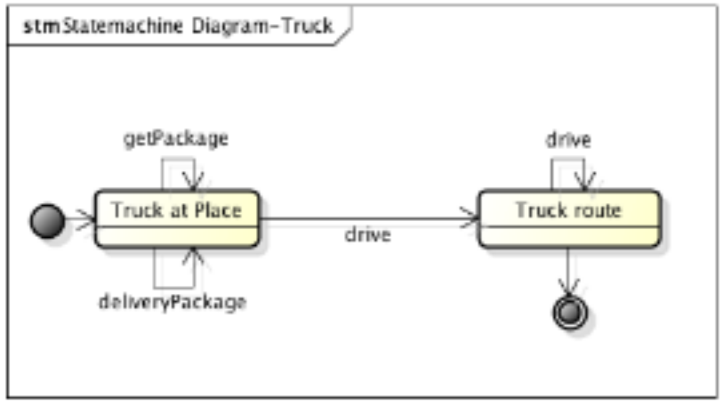


Figure 4: State Diagram designed to represent the planning problem.

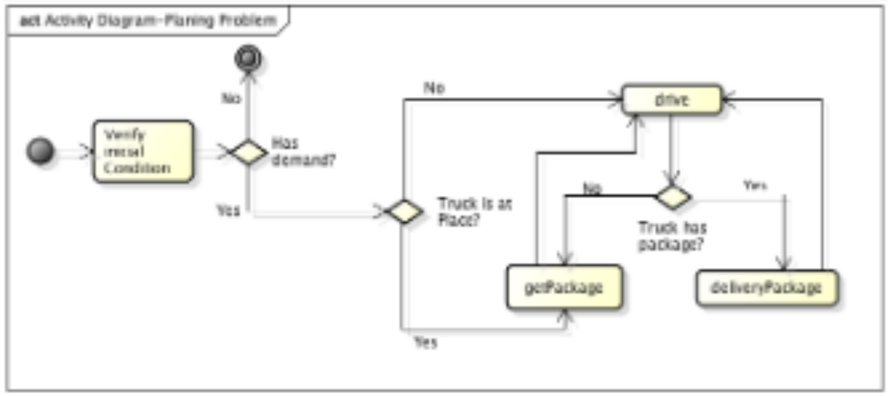


Figure 5: Activity Diagram designed to represent the planning problem.

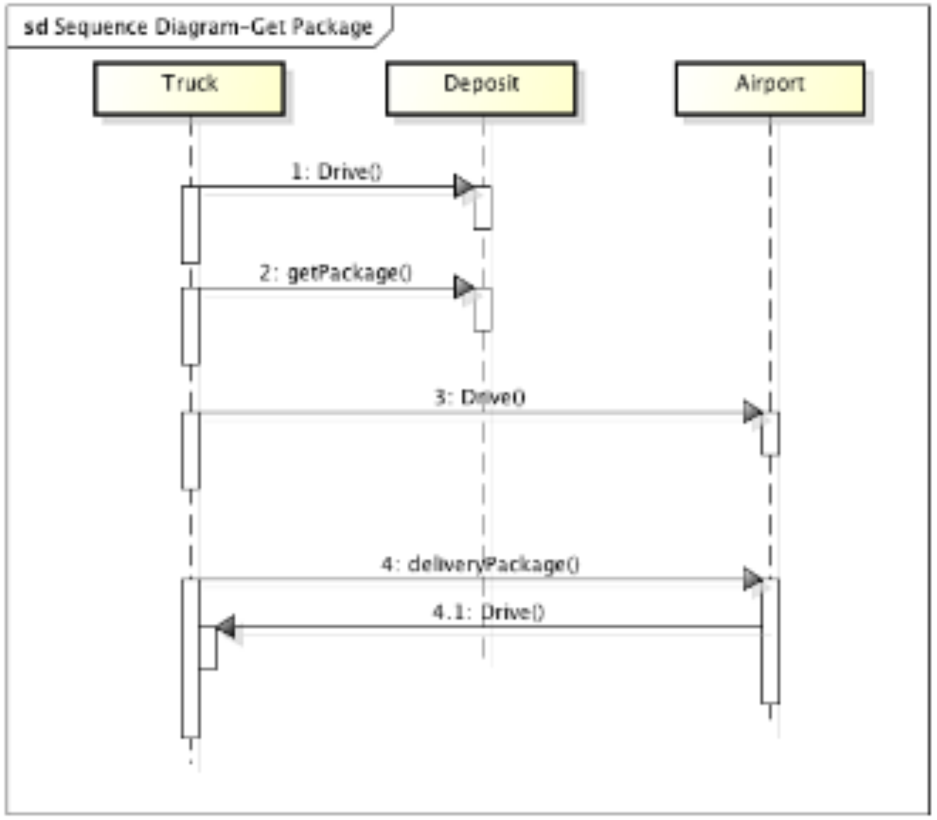
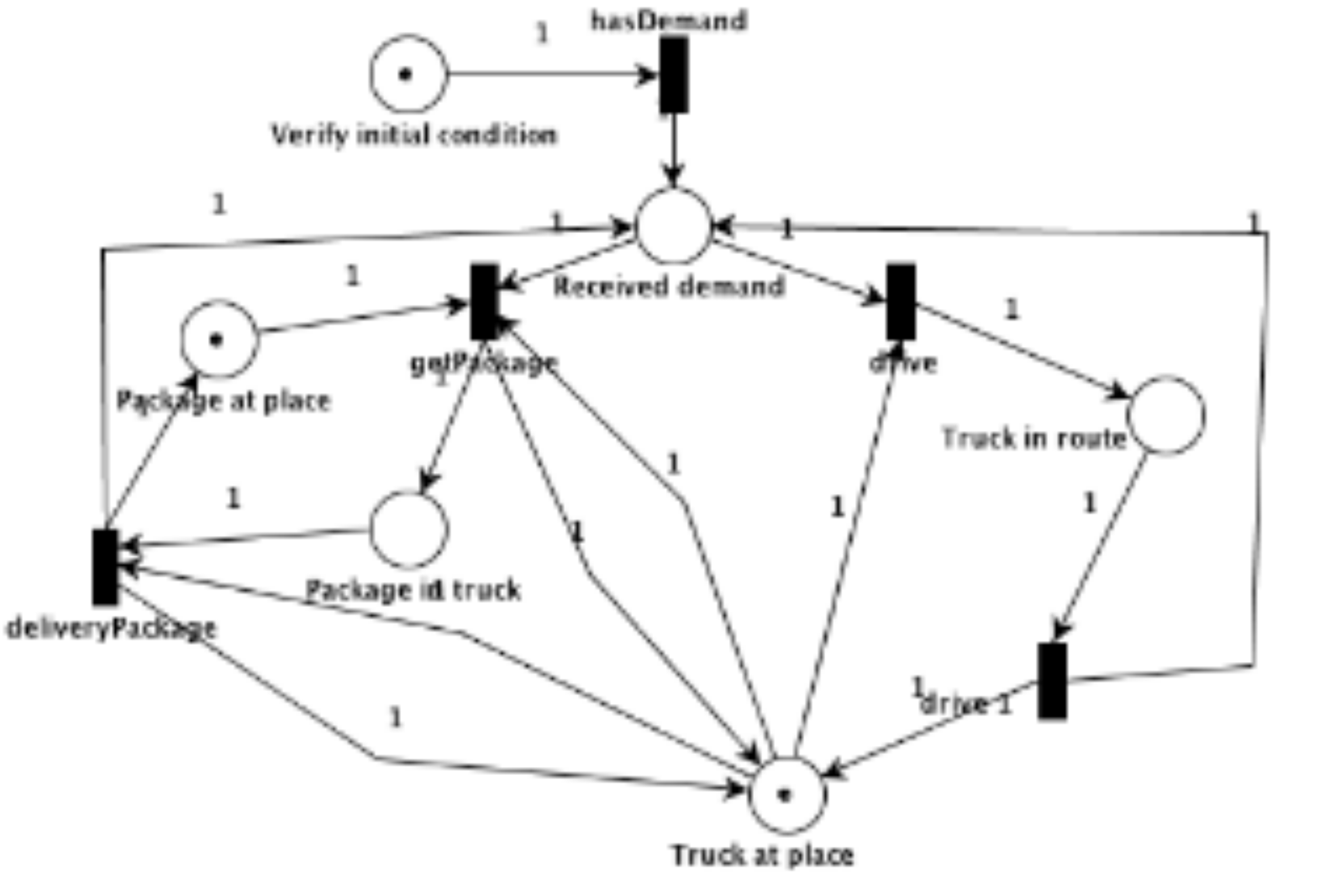


Figure 6: Sequence Diagram designed to represent the planning problem.



XIII Simpósio Brasileiro de Automação Inteligente
Porto Alegre – RS, 1º – 4 de Outubro de 2017

GORE METHODS TO MODEL REAL WORLD PROBLEM DOMAINS IN AUTOMATED PLANNING

Javier Martínez Silva*, José Reinaldo Silva*

*Department of Mechatronics Engineering, University of São Paulo, São Paulo, Brazil

Emails: javsilva@usp.br, reinaldo@usp.br

Abstract— In the field of intelligent design, the early phase, dedicated to requirements modeling and analysis, plays a fundamental role, especially when analytic formal solutions are not suitable. Automated planning falls in that category - particularly when the target are "real world" systems. In requirement analysis Knowledge Engineering is explored to provide clues that can facilitate a convergence for a good planning solution. Therefore, a great effort has been made today in the area of Artificial Intelligence to define a reliable design process for automated planning that includes Knowledge Engineering treatment in the early phase, coupled to requirements modeling and analysis. This paper presents an integrated approach to requirements analysis based on GORE (Goal Oriented Requirements Engineering) that starts by modeling a knowledge architecture based on a domain and planning requirements represented in KAOS and converted in Petri Nets (PN) to analysis. A software tool called rekPlan is proposed to generate the PN graph. The analysis is made in another software tool proposed in our Lab that, GHENESys (General Hierarchical Enhanced Net System), that support unified Petri Nets following ISO/IEC 15.909. A real case study is presented, based on classic problems of pre-salt petroleum industries.

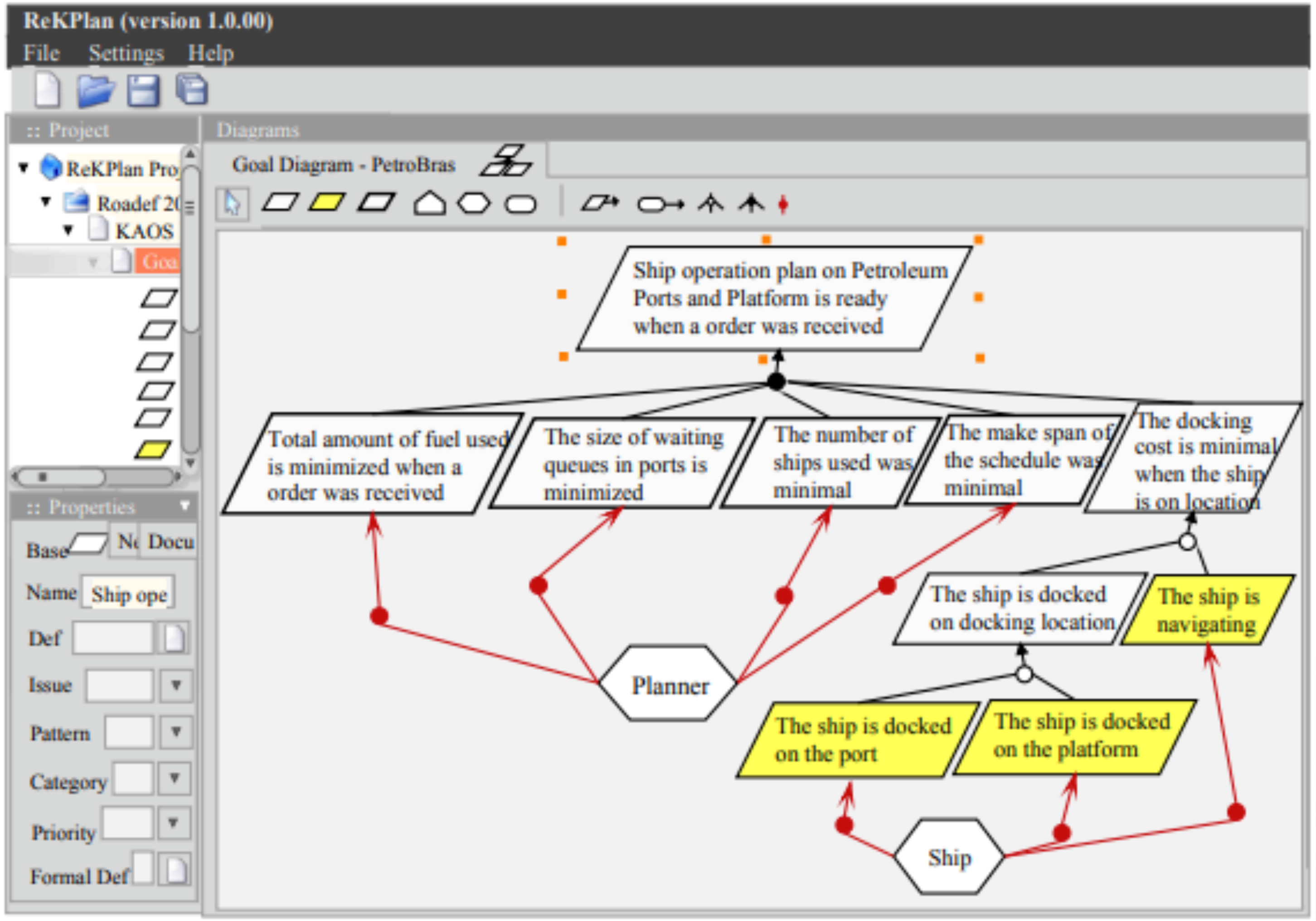
Keywords— Requirement Engineering, GORE, Petri Net, Automated Planning

1 Introduction

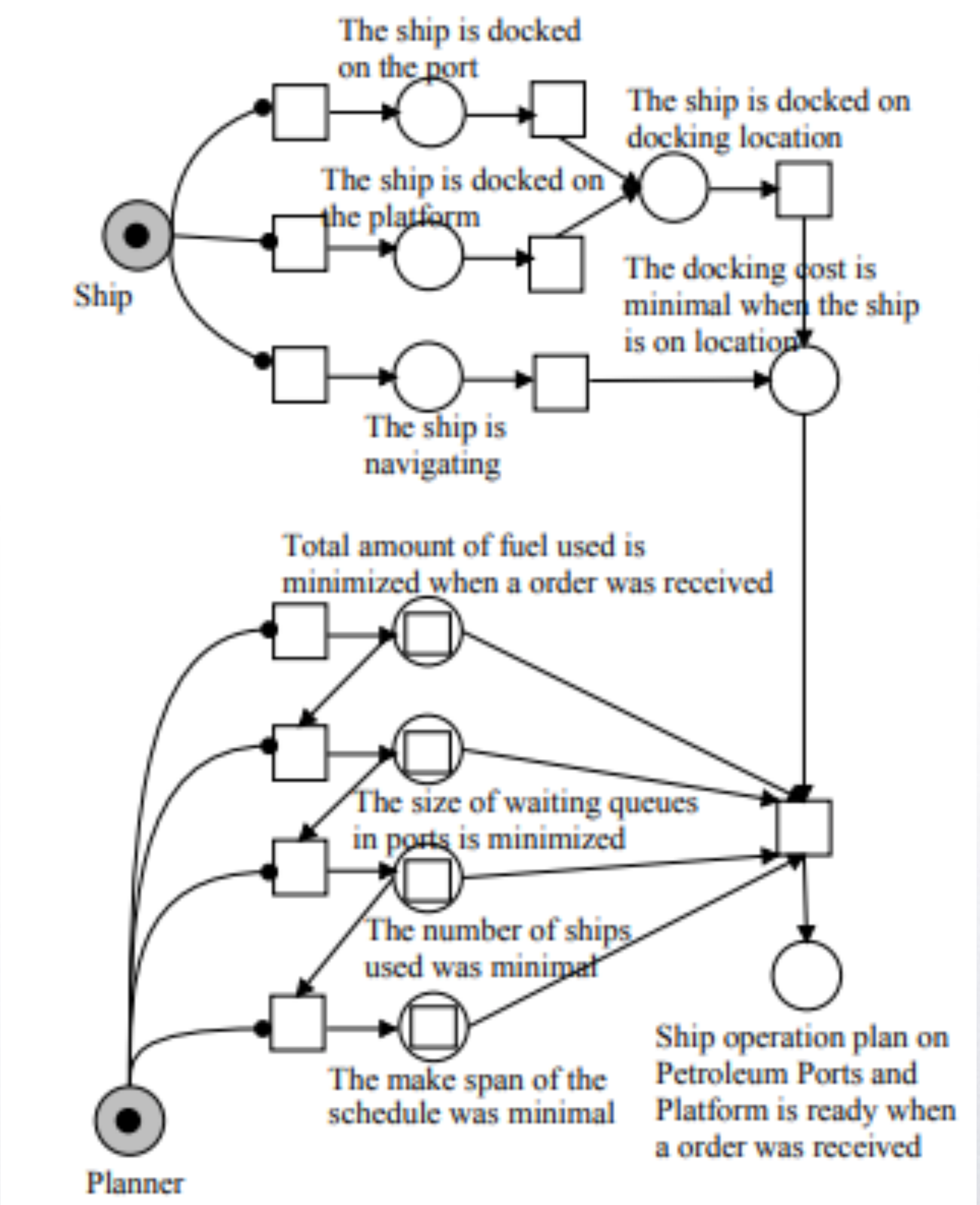
Planning characterizes a specific type of design problem where the purpose is to find an admissible sequence of actions to bring the system from a given initial state to a target final state. Current approaches in the literature aim to improve the performance of automated planners by trying to optimize the search algorithms and the general solution (Lipovetzky and Geffner, 2017). In addition, most existing work on this direction is conceptually tested in synthesized artificial problems (closed problems that have limited set of actions) as. On the other hand, due to the extensive development in this area, some authors started to apply planning techniques to real world problems as well - like logistic problems - with a considerable higher number of variables, where the classic domain independent approaches are computationally prohibitive (Vaquero et al., 2012). Such alternative approach could bring some light and/or good results to challenge problems and could also gave some feedback to solve a fully automated, domain-independent problem.

tics problem proposed in ROADEF has a map of cities connected by airline routes. Transportation inside cities uses a truck (there is one truck in each city). Cities are abstracted, being treated as destination points. Inside a city, a truck can go from any point to any destination at no cost (Lin et al., 2016). However, in the real world, transportation within a city is a sub-problem that can involve considerable costs.

This paper intents to propose a requirement analysis formal procedure, based on hierarchical models, that starts by taking requirements of planning problems represented in KAOS (Keep All Object Satisfied) and proceed to analysis based on classic Petri Nets. This approach were inserted in a knowledge based tool called reKPlan (Requirement Engineering based on KAOS for Planning Problems) framework, that performs the KAOS/GHENeSys net translation (and eventually translates from GHENeSys to Linear Temporal Logic, LTL). Thus, the KAOS diagram is translated to a Petri net through a transference algorithm, proposed by (MARTINEZ SILVA, 2016)



ReKPlan (version 1.0.00)
File Settings Help
:: Project
Diagrams
Goal Diagram - PetroBras
ReKPlan Pro
Roadef 2
KAOS
:: Properties
Base
Name Ship ope
Def
Issue
Pattern
Category
Priority
Formal Def
Ship operation plan on Petroleum Ports and Platform is ready when a order was received
Total amount of fuel used is minimized when a order was received
The size of waiting queues in ports is minimized
The number of ships used was minimal
The make span of the schedule was minimal
The docking cost is minimal when the ship is on location
The ship is docked on docking location
The ship is navigating
Planner
The ship is docked on the port
The ship is docked on the platform
Ship



The ship is docked on the port
The ship is docked on docking location
The ship is docked on the platform
Ship
The docking cost is minimal when the ship is on location
The ship is navigating
Total amount of fuel used is minimized when a order was received
The size of waiting queues in ports is minimized
The number of ships used was minimal
The make span of the schedule was minimal
Ship operation plan on Petroleum Ports and Platform is ready when a order was received
Planner

d-l@b

MECATRÔNICA

Engineering Applications of Artificial Intelligence 81 (2019) 373–386

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

ELSEVIER

A new hierarchical approach to requirement analysis of problems in automated planning ☆

Check for updates

Javier Martinez Silva *, José Reinaldo Silva

Department of Mechatronics Engineering, University of São Paulo, Brazil

ARTICLE INFO

Keywords:
Knowledge engineering
Hierarchical automated planning
Planning domain analysis
Hierarchical petri nets
Requirements analysis

ABSTRACT

The use of Knowledge Engineering (KE) processes to analyze and configure domains in automated planning is becoming more appealing since it was noticed that this issue could make a difference to solve real problems. The contrast between a generic domain independent approach, taken as canonical in AI, and alternative processes that include knowledge engineering – eventually adding specific knowledge – has been discussed by Computer and Engineering communities. A big impact has been noticed mainly in the early phase of requirement analysis when KE approach is normally introduced. Requirement analysis is responsible for carrying out the Knowledge modeling of both problem and work domains, which is a key issue to guide different planner algorithms to come out with efficient solutions. Also, there is the scalability issue that appear in most real problems. To face that, hierarchical methods played an important hole in the history of planning and inspired several solutions since the proposal of NONLIN in the 70's. Since then, the idea of associating hierarchical relational nets with partial ordered actions has prevailed when large systems were considered. However, there is still a gap between the hierarchical approach and the state of art of requirements analysis to allow features anticipated by KE approach to really appear in the requirements of a planning process. This paper proposes a pathway to solve this gap starting with requirements elicitation represented first in the conventional semi-formal (diagrammatic) language – UML – that is translated to Hierarchical Petri Nets (HPNs) by a new enhanced algorithm. The proposed process was installed in a software tool – developed by one of the authors – that analyzes the performance of the KE planning model: itSIMPLE (Integrated Tools Software Interface for Modeling Planning Environment). This tool was initially designed to use classic Place/Transition nets and an old version of UML (2.1). It is now enhanced to use UML 2.4 and a hierarchical Petri Net extension, also developed by the authors. Realistic examples illustrate the process which is now being applied to larger problems related to the manufacturing of car sequencing domain, one of challenge of ROADEF 2005 (French Operations Research & Decision Support Society). Finally, we consider the possibility to introduce another approach to the KE process by using KAOS (Keep All Object Satisfied) to make the planning design more accurate.

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.

Fim