SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UFC UNIVERSITÉ DE FRANCHE-COMTÉ

N° 0 8 2

THÈSE présentée par

HADRIEN BRIDE

pour obtenir le

Grade de Docteur de
l'Université de Franche-Comté

Spécialité : **Informatique**

Verifying Modal Specifications of Workflow Nets

Using constraint solving and reduction methods

Soutenue publiquement le 24 Octobre 2016 devant le Jury composé de :

DIDIER BUCHS	Raporteur	Professeur à l'Université de Genève, Suisse
CLAUDE JARD	Raporteur	Professeur à l'Université de Nantes, France
SERGE HADDAD	Examinateur	Professeur à l'École normale supérieure de Cachan, France
CATHERINE DUBOIS	Examinateur	Professeur à l'Ecole Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise, France
OLGA KOUCHNARENKO	Directeur de thèse	Professeur à l'Université de Franche-Comté, France
FABIEN PEUREUX	Co-encadrant de thèse	Maître de conférence à l'Université de Franche-Comté, France

Contents

I Introduction and State of the Art 1

1 Introduction 3

1.1 Context and Motivations . . . 3
1.2 Research Questions . . . 6
1.3 Contributions to this Thesis . . . 6
1.4 Outline . . . 7

2 State of the Art 9

2.1 Preliminaries . . . 9
2.1.1 Notions of Set Theory and Graph Theory . . . 9
2.1.2 Petri Nets . . . 11
2.1.2.1 Matrix Representation of Petri nets . . . 14
2.1.3 Petri Nets Behavioural Properties . . . 15
2.1.4 Petri Nets Subclasses . . . 16
2.1.5 Petri Net Extensions . . . 18
2.1.5.1 Hierarchical Petri Nets / Refinement . . . 18
2.1.5.2 Coloured Petri Nets . . . 19
2.1.5.3 Weighted Transitions Petri Nets . . . 21
2.1.6 Workflow Nets . . . 21
2.1.7 Workflow Nets Equivalences . . . 23
2.1.8 Workflow Nets Soundness . . . 24
2.1.9 Stepwise Refinement of Worflow Nets . . . 25
2.1.10 Modal Specification . . . 28
2.2 Analysis of Petri Nets . . . 28
2.2.1 Behavioural Approach . . . 29
2.2.2 Structural Approach . . . 30
2.2.2.1 State Equation . . . 30
2.2.2.2 Structural Invariants . . . 31
2.2.2.3 Siphons and Traps . . . 33

2.2.2.4 Analysis of Workflow Nets 34
2.2.3 Reduction Approach 35
2.2.3.1 Formalization 35
2.2.3.2 Well-known reduction rules 36
2.3 Constraint Systems 38
2.3.1 Definition 38
2.3.2 Constraint Logic Programming (CSP) 39
2.3.3 Satisfiability Modulo Theories (SMT) 39

II Contributions 41

3 Verification of Modal specification 43
3.1 Over Ordinary Workflow Nets 44
3.1.1 Extended Modal Specification 44
3.1.2 Modelling Executions of Workflow Nets 46
3.1.3 Verifying Extended Modal Specifications 53
3.2 Over Abstract Workflow Nets 55
3.2.1 Abstract Petri Nets 55
3.2.2 Extended Abstract Modal Specification 59
3.2.3 Modelling Executions of Abstract Workflow Nets 61
3.2.4 Verifying Extended Abstract Modal Specifications 64
3.3 Synthesis 65

4 Reduction methods 67
4.1 Φ^*: A workflow nets reduction kit 68
4.2 Semi-Decision of Generalised Soundness 77
4.3 Preprocessing Modal Specification Verification 79
4.3.1 Reduction based on hierarchical workflow nets 80
4.3.2 Reduction based on reduction rules 82
4.4 Synthesis 86

5 Experimental Evaluation 89
5.1 Study Cases 90
5.1.1 Issue Tracking System 90
5.1.2 Question and Answer Portal 91
5.1.3 Tax Accounting Manager 95

5.2 Tool Chain Implementation 98
5.2.1 Modal Specification Verifier 98
5.2.2 Reduction Tool 103
5.3 Study cases results 104
5.4 Scalability 108
5.4.1 Benchmark's Generation Tool 108
5.4.2 Experimental Evaluation of Modal Specification Verification 109
5.4.2.1 Objectives 110
5.4.2.2 Experimental Protocol 110
5.4.2.3 Results and Feedback from Experiments 111
5.4.3 Experimental Evaluation of Reduction Methods 117
5.4.3.1 Objectives 118
5.4.3.2 Experimental Protocol 118
5.4.3.3 Results and Feedback from Experiments 119
5.5 Synthesis 123

III Conclusion and Future Work 125

6 Conclusion 127
6.1 Verification of Modal specifications 127
6.2 Reduction methods 128
6.3 Experimental Evaluation 129

7 Future Work 131
7.1 Towards Parallelism 131
7.2 Error-pattern 132
7.3 Reconfiguration 133

I

Introduction and State of the Art

1

Introduction

Contents

1.1 Context and Motivations 3
1.2 Research Questions 6
1.3 Contributions to this Thesis 6
1.4 Outline 7

1.1/ Context and Motivations

Our ability to attain fitness goals through complexly organized, situationally-tailored, instrumental sequences of behaviours is one of the defining trait of our human nature [DeVore et al., 1987]. Human progress is essentially based on the improvement of the processes designed to attain fitness goals (e.g., survive, reproduce). In modern times, the organisation of such processes into logically coherent systems, called rationalization, has led to major improvements. For example, the rationalization of manufacturing processes has promoted the industrial revolution, an important and beneficial mark of progress [Ashton et al., 1997]. In a broader context, the rationalization of goal-oriented processes gave raise to the notion of workflows.

Intuitively, a workflow describes the set of possible runs of a particular process by specifying the series of activities that are necessary to complete it. More precisely, the Workflow Management Coalition [Coalition, 1996], a global organization of adopters, developers, consultants, analysts, as well as university and research groups engaged in workflow-based development, defined workflow as: *The automation of a process, in whole or part, during which documents, information or tasks are passed from one participant (i.e. a person or an automated process) to another for action (i.e. activities), according to a set of procedural rules.*

Nowadays workflows are extensively used by companies and organisations in order to improve organizational efficiency, responsiveness and profitability by managing the tasks and steps of business processes [Schäl, 1996, Lawrence, 1997, Fischer, 2003, Van Der Aalst et al., 2004].

To effectively use and manage workflows, companies rely more and more on workflow management systems which provide an infrastructure for the set-up, execution and monitoring of a defined sequence of tasks, arranged as a workflow. A great diversity of application domains exist today that use workflow management systems on a daily basis in order to control their business processes. These include office automation, healthcare, telecommunications, manufacturing and production, finance and banking, just to name a few.

Workflows modelling is one of the most important steps in managing workflows. It aims at mapping out the different activities involved in processes so that they can be understood, evaluated, and improved. On the one hand, well designed workflows allow business processes to become more efficient, compliant, agile and visible by ensuring that every process step is explicitly defined and optimized for maximum productivity. On the other hand, faulty workflows often result in financial losses as well as in failure to provide services.

With the increasing use of workflows for modelling crucial business processes, the verification of specifications, i.e. of desired properties of workflows, becomes mandatory to ensure that such processes are properly designed and reach the expected level of trust and quality. For this reason, in the last two decades, companies have been putting more stress on the analysis of their business processes in order to maintain a competitive level while complying with a high quality of services.

Workflow analysis can be carried out by informal approaches such as workshops in which the interested stakeholders can discuss and point to possible issues in the defined workflows. However, the growing complexity of the modelled processes requires new advanced methods and dedicated analysis tools to achieve automation, reliability as well as scalability of the workflow analysis and verification steps.

Many workflow modelling languages have been proposed to model workflows (e.g., UML activity diagrams [Dumas et al., 2001], BPMN [White, 2004], Event-driven Process Chains [Scheer et al., 2005]). Among them, workflow nets [van der Aalst, 1998], a particular class of Petri nets [Petri, 1962], have become one of the standard ways to model and analyse workflows. A workflow net is typically used as an abstraction of a workflow, notably modelling its control-flow dimension. The success of modelling workflows as Petri nets can be explained by several reasons. First, Petri nets are a graphical language, as a result they are intuitive and easy to learn. Second, Petri nets are very expressive: they allow the modelling of complex workflows exhibiting concurrencies, conflicts, as well as causal dependencies of activities (i.e tasks). Finally and most importantly, Petri nets have a clear and precise formal definition of their semantics. This enables their validation and verification within the framework of formal methods [Clarke et al., 1996], a set of mathematically based languages, techniques, and tools for specifying and verifying such systems.

While most workflow nets verification problems are known decidable [Esparza, 1998], their complexity is often very high due to the large expressiveness provided by Petri nets. For instance, the reachability problem (i.e. the problem of determining whether a given state is reachable), a central problem to which many others reduce to, is known to be decidable. However, it requires exponential space and therefore exponential time [Lipton, 1976].

To assist engineers in their specification and validation activities, modal specifications [Larsen, 1989] have been designed to allow *loose* specifications with restrictions on transitions of complex systems. Those specifications are notably used within refinement approaches for workflows development, a top-down design approach to workflows modelling which consists in iteratively (i.e. step by step) refining a model by adding further details until a sufficient level of description is obtained. More precisely, in the context of workflow specification, modal specifications allow the definition of *necessary* or *admissible* behaviours to which the considered workflow model must conform. Modal specifications are usually expressed over a single transition of the considered system. As pointed out to us by workflow modellers of the study cases considered in this thesis, this is quite limiting. The issue is that, in real-life, the expression of complex modal behaviours involving several transitions and their causalities is needed. This thesis addresses this issue and presents extended modal specifications – a modal logic expressing complex modal behaviour involving several transitions and dealing with their causalities.

The verification of behavioural properties, such as modal specifications, are usually performed through model checking [Clarke et al., 1999]. Model checking is a verification technique that explores all possible system states in a brute-force manner. It aims at verifying the validity of a desired behavioural property (the specification) via an exhaustive – explicit or symbolic – enumeration of all the reachable states and transitions between them. The complexity of such an approach can be worse than primitive recursive space due to the state explosion problem [Valmari, 1998]. This renders model checking and other techniques based on state space exploration intractable on large instances of workflow nets.

On the one hand, verifying behavioural properties of workflows is a very complex task which requires exponential computational resources with respect to the size of their modelling by workflow nets. On the other hand, the size of companies workflows is growing at a very fast pace as companies become larger, more complex, and rely more and more on workflow management systems to deal with an evermore precise and detailed range of business processes.

The main challenge faced during the development of verification approaches is therefore this inerrant complexity with respect to the size of the considered workflow nets. Indeed, complete and efficient verification tools for the general case are currently out of reach. However, to cope with this issue, different approaches have been considered. A first one consists in considering less expressive subclasses of workflow nets for which efficient verification procedures can be designed [Van der Aalst, 1997]. This method has been thoroughly investigated [van der Aalst et al., 2011] and is based on compromises between expressiveness of the workflow models and the complexity of their analysis. While large subclasses of workflow nets with manageable analysis complexity have been identified (e.g., free-choice workflow nets), this approach strongly limits the range of workflow behaviour that can be modelled. A second approach consists in designing verification heuristics. Such procedures (e.g., [Vanhatalo et al., 2007, Wimmel et al., 2011]) aim to efficiently handle a large range of usual workflow nets by guiding the explicit or implicit state space exploration. These procedures, which are not necessary complete, have shown their efficiency in operational contexts [Vanhatalo et al., 2007, Wimmel et al., 2011, Esparza et al., 2014, Esparza et al., 2015]. They are often based on the design of abstract models approximating the considered workflow net behaviour. The construction of such abstract models relies primarily on powerful abstraction mechanisms. This thesis addresses the verification of extended modal specifications in line with this second approach.

Recent advances in the development of constraint solving tools offer an unprecedented opportunity for the efficient automation of analysis tasks [Prasad et al., 2005, Rybalchenko, 2010]. Existing constraint solvers present a real opportunity to leverage these advances for solving hard program analysis problems [Gulwani et al., 2008] such as the problems faced during workflow nets analysis. Constraint-based algorithms are composed of two main steps. In a first step, the validity of a property of interest is formulated as a constraint system (i.e. a set of constraints over a set of variables ranging over their domains) through an automated process, called constraint generation. In a second step, this constraint system is solved to determine the validity of the considered property. This second step is executed using a third party constraint solver. Such a separation of concerns – constraint generation versus constraint solving – can liberate the designer of the verification tool from the tedious task of creating a dedicated algorithm by beneficing from the maturity and efficiency of existing constraint solvers.

On the basis of the above, this thesis presents contributions made to the verification of modal specifications of workflow nets based on powerful abstractions and the use of constraint solving. The research questions we will address are summarized in the next section.

1.2/ Research Questions

As previously introduced, this work is motivated by the challenges raised during the analysis of workflow described by workflow nets. More precisely, it considers the following research questions:

RQ_1 *How to effectively express modal behaviour of workflow nets involving several transitions and their causalities*?

In the context of workflow nets, modal specifications are usually expressed over a single transition of the system under analysis. This is quite limiting as the expression of complex modal behaviour involving several transitions and their causalities is needed in real-life.

RQ_2 *How to effectively verify the validity of such modal specifications by leveraging the efficiency of existing mature constraint solvers?*

The design of a constraint system based framework enabling the verification of modal specifications, which express complex modal behaviours involving several transitions and their causalities, will benefit from the use of existing mature and efficient constraint solvers.

RQ_3 *Which abstraction mechanisms are appropriate and relevant with respect to verification of such modal specifications?*

As stated in the previous section, the verification of modal specifications is a hard verification problem. Due to its inerrant complexity with respect to the size of the considered workflow nets, powerful abstractions preserving the validity of modal specifications are needed to efficiently handle a large range of usual workflow nets.

1.3/ Contributions to this Thesis

This section briefly describes the contributions made to this thesis in order to address the research questions introduced in the previous section.

To address RQ_1, we first define a modal logic, over workflow nets, enabling the description of modal behaviour involving several activities and their causalities [Bride et al., 2015]. This modal logic stems from the observation that while basic modal specifications are useful, they usually lack expressiveness for real-life applications, as only individual transitions are concerned with. This modal logic is then further extended to handle workflow nets extensions (e.g., workflow nets with data). To this end, additional constraints on the initial and final states as well as on the data associated to transition firing are considered [Bride et al., 2015].

In a second step, to address RQ_2, we describe a novel constraint-based framework for modelling workflow nets executions. In this framework, several over-approximations of the correct executions of workflow nets are defined as constraint systems. We also define a constraint system whose solution space under-approximates the set of correct executions of workflow nets. Solutions of this latter constraint system are called segments. They model partial executions whose existence is guaranteed through structural analysis. Furthermore, we show that the concatenation of such segments can be used to model any correct executions of workflow nets. This constraint-based framework is used to verify the validity of modal specifications expressed within the previously defined modal logic by leveraging the efficiency of mature constraint solvers. These results have been published in [Bride et al., 2014]. We then proceed to explicitly describe how our modal

specification description and verification methodology can be applied to a more abstract notion of workflow nets that we define. This abstract notion of workflow nets notably includes workflow nets with data on which the proposed approach can be carried out as published in [Bride et al., 2015].

In a third step, to address RQ_3, we portray reduction methods based on powerful abstraction mechanisms in order to reduce the size of analysed workflow nets while preserving the behavioural properties of interest. To this end, we present workflows nets reduction rules strongly preserving generalised soundness, an essential and necessary correctness property that must be satisfied by workflow nets. The presented reduction rules extend existing reduction rules and therefore enable a greater reduction of workflow nets. We show how they are used as powerful pre-processing steps to the verification of generalised soundness as well as to the verification of modal specifications.

Finally, as a practical contribution to RQ_2 and RQ_3, these approaches have been implemented and experimentations have been carried out in order to validate them. We introduce and discuss convincing experimentations carried out over real-life industrial study cases in order to illustrate the relevance and effectiveness of the proposed modal specification verification approach. Furthermore, we present an empirical evaluation of effectiveness, efficiently and scalability of the proposed modal specification verification approach over workflow nets of growing size and complexity. This experimental work has been published in [Bride et al., 2016a, Bride et al., 2016b]. Finally, we also describe experimental results supporting the benefits provided by the presented reduction methods to the verification of workflow net behavioural properties such as generalised soundness and correctness with respect to modal specifications.

1.4/ Outline

This thesis dissertation is organised according to the following outline.

In Chapter 2, we provide a review of some mathematical notions and notations used throughout this thesis. We also review some elements of the state of the art regarding Petri nets, workflow nets and existing approaches for their analysis as well as constraint systems together with approaches for their resolution.

In Chapter 3, we define extended modal specifications, a modal logic which extends usual modal specifications by enabling the description of *necessary* or *admissible* behaviour involving several activities and their causalities. We then, in a first step, define an innovative constraint system based framework to model executions of workflow nets. This framework is then used to verify extended modal specifications. In a second step, we define abstract workflow nets, an abstract notion of workflow nets which notably provides a generalisation of Petri nets, and of coloured Petri nets. The previously defined framework based on constraint systems is then applied to such an abstract notion of workflow nets to provide an extended modal specification verification method to analyse not only ordinary workflow nets but also coloured workflow nets and similar extensions.

In Chapter 4, powerful reduction methods preserving properties of interest such as generalised soundness and correctness of a given extended modal specification are presented. We then portray pre-processing steps based on these reduction techniques reducing workflow nets size, so that the analysis of preserved properties can be carried out on smaller instances.

In Chapter 5, we present dedicated tools implementing the approaches defined in Chapters 3 and 4, and we detail and discuss experimental results obtained over real-life industrial study cases and benchmarks in order to assess their value.

Chapters 6 and 7 respectively draw general conclusions and gives directions for future work.

2

State of the Art

"Those who don't know history are doomed to repeat it."

— Edmund Burke

Contents

2.1 Preliminaries 9
2.1.1 Notions of Set Theory and Graph Theory 9
2.1.2 Petri Nets 11
2.1.3 Petri Nets Behavioural Properties 15
2.1.4 Petri Nets Subclasses 16
2.1.5 Petri Net Extensions 18
2.1.6 Workflow Nets 21
2.1.7 Workflow Nets Equivalences 23
2.1.8 Workflow Nets Soundness 24
2.1.9 Stepwise Refinement of Worflow Nets 25
2.1.10 Modal Specification 28
2.2 Analysis of Petri Nets 28
2.2.1 Behavioural Approach 29
2.2.2 Structural Approach 30
2.2.3 Reduction Approach 35
2.3 Constraint Systems 38
2.3.1 Definition 38
2.3.2 Constraint Logic Programming (CSP) 39
2.3.3 Satisfiability Modulo Theories (SMT) 39

2.1/ Preliminaries

2.1.1/ Notions of Set Theory and Graph Theory

This section reviews basics notions/definitions about set theory and graph theory. For a thorough description we refer the interested reader to [Jech, 2013, West et al., 2001].

Set Theory

Set theory is the branch of mathematical logic that studies sets (i.e. collections of objects) commonly employed as a foundational system for mathematics, particularly in the form of Zermelo–Fraenkel set theory [Hayden et al., 1968] with the axiom of choice.

Intuitively, a set is a collection of distinct objects. Throughout this manuscript, standard notation for sets and operations are used.

The *empty set* is denoted by $\emptyset$, *element inclusion* (i.e. element membership) by $\in$, *set intersection* by $\cap$, *set union* by $\cup$, *set difference* by $\setminus$, *set inclusion* by $\subseteq$, and *strict set inclusion* by $\subset$. We denote the *set of natural numbers* by $\mathbb{N}$, and the *set of integers* by $\mathbb{Z}$.

The number of elements in a finite set A is called the cardinality, and is denoted by $|A|$.

The Cartesian product allows a new set to be created from existing sets.

Definition 1: Cartesian Product

The Cartesian product of two sets A and B, denoted $A \times B$, is the set of ordered pairs $\{(a, b) \mid a \in A, b \in B\}$.

Correspondences between two sets are defined by binary relations.

Definition 2: Binary Relation

A binary relation R over two sets A and B is a subset of $A \times B$. The *domain* of R, denoted $Dom(R)$, is the set $\{a \in A \mid \exists\, b \in B, (a, b) \in R\}$. The *co-domain* (also called the range) of R, denoted $Ran(R)$, is the set $\{b \in B \mid \exists\, a \in A, (a, b) \in R\}$. The notation aRb signifies that $(a, b) \in R$.

A binary relation $f \subseteq A \times B$ is called a *partial function*, denoted $f : A \nrightarrow B$ if $\forall\, (a_1, b_1), (a_2, b_2) \in f, a_1 = a_2 \Rightarrow b_1 = b_2$. The notation $f(a) = b$ signifies that $(a, b) \in f$. A *partial function* $f : A \nrightarrow B$ is called a *function*, denoted by $f : A \rightarrow B$, if $Dom(A) = A$.

An enumerated collection of objects in which repetitions are allowed is called a sequence.

Definition 3: Sequence

A sequence S is an enumerated collection of objects from a set A in which repetitions are allowed. It is defined as a function $f_S : D \rightarrow A$ where $D \subseteq \mathbb{N} \setminus \{0\}$. By convention, for $i \in D$, the i^{th} element of S is denoted $S_i = f_S(i)$.

A binary operation $\odot$ over a set A is a *function* $\odot : A \times A \rightarrow A$ that combines two elements of A (called operands) to produce another element of A. By convention, let $a_1, a_2 \in A$, we denote $\odot(a_1, a_2)$ by $a_1 \odot a_2$. The binary operation $\odot$ is said to be an associative binary operation if and only if $\forall\, a_1, a_2, a_3 \in A, (a_1 \odot a_2) \odot a_3 = a_1 \odot (a_2 \odot a_3)$ and is said to be a commutative binary operation if and only if $\forall\, a_1, a_2 \in A, a_1 \odot a_2 = a_2 \odot a_1$.

A monoid is a set that has an identity element together with an associative binary operation.

Definition 4: Monoid

Let A be a set and $\odot$ be a binary operation over A.
$(A, \odot)$ is a monoid if and only if:

- $\odot$ is an associative binary operation, and
- there exists $0_A \in A$, an identity element, such that $\forall\, a \in A, a \odot 0_A = 0_A \odot a = a$.

A commutative monoid is a monoid whose associative binary operation is commutative.

For example, the natural numbers $\mathbb{N}$ form a commutative monoid under addition (identity element zero), or multiplication (identity element one).

Any commutative monoid $(A, \odot)$ is endowed with its algebraic preordering $\leq_A$ defined by $\forall\, a_1, a_2 \in A, a_1 \leq_A a_2 \Leftrightarrow \exists\, a_3 \in A, a_1 \odot a_3 = a_2$.

Graph Theory

A set together with a binary relation over itself forms a directed graph.

Definition 5: Directed Graph

A directed graph G is a tuple $\langle N, A \rangle$ where N is a set of nodes and $A \subseteq N \times N$ is a set of directed arcs.

A directed graph whose set of nodes can be divided into two disjoint sets such that directed arcs do not relate two nodes of the same set, is called a bipartite directed graph.

Definition 6: Bipartite Directed Graph

A bipartite directed graph G is a directed graph $\langle N_1 \cup N_2, A \rangle$ where $N_1 \cap N_2 = \emptyset$ and $A \subseteq N_1 \times N_2 \cup N_2 \times N_1$.

A sequence of directed arcs which connect a sequence of nodes is called a path.

Definition 7: Path

Let $G = \langle N, A \rangle$ be a directed graph and $n \in \mathbb{N}$.
A path $\sigma : \{1, .., n\} \to N$ of length n in G is a finite sequence of nodes such that $\forall\, i \in \{1, .., n-1\}, (\sigma(i), \sigma(i+1) \in A$.

2.1.2/ Petri Nets

Petri nets, also known as place/transition nets, are a basic model of parallel and distributed systems proposed by Carl Adam Petri [Petri, 1962]. They allow modelling of discrete event systems exhibiting behaviours such as concurrency, conflict, and causal dependency between events in a readable graphical and/or a formal manner, and several important verification problems, like reachability or soundness, are known to be decidable [Esparza, 1998]. They are widely used to model concurrent processes in theoretical computer science. They are also used to describe chemical reactions, manufacturing processes, supply chains, and so on.

A Petri net is a directed bipartite graph, in which the nodes represent transitions (i.e. events that may occur, graphically depicted by bars) and places (i.e. conditions, graphically represented by circles). A transition (i.e. an event) may have any numbers of input places and output places which respectively represent the pre-condition and the post-condition of the considered transition.

Formally an ordinary Petri net is defined as follows.

Definition 8: Ordinary Petri net

An ordinary Petri net N is a tuple $\langle P, T, F \rangle$ where:

- P is a finite set of places,
- T is a finite set of transitions ($P \cap T = \emptyset$),
- $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs, also called the flow relation of N.

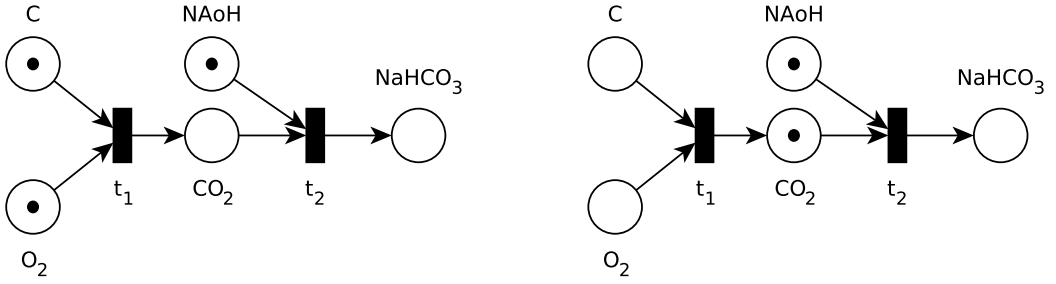
The nodes from which an arc runs to a node g are called the input nodes of g and are denoted by ${}^\bullet g$. Likewise, the nodes to which arcs run from a node g are called the output nodes of g and are denoted by $g^\bullet$. Formally, let $g \in P \cup T$ and $G \subseteq P \cup T$ we have:

$$g^\bullet = \{g' | (g, g') \in F\} \qquad {}^\bullet g = \{g' | (g', g) \in F\}$$

$$G^\bullet = \cup_{g \in G} g^\bullet \qquad {}^\bullet G = \cup_{g \in G} {}^\bullet g$$

A *marking* of a Petri net, representing the number of tokens on each place, is a function $M : P \to \mathbb{N}$. It evolves during its execution since transitions change the marking of a Petri net according to the following *firing rules*. A transition t is *enabled* in a marking M_a if and only if $\forall p \in {}^\bullet t, M_a(p) \geq 1$. When an enabled transition t is *fired*, it *consumes* one token from each place of ${}^\bullet t$ and *produces* one token for each place of $t^\bullet$. Formally, the firing of a transition t, enabled in a marking M_a, results in a new marking M_b defined as follows:

$$\forall\ p \in P,\ M_b(p) = \begin{cases} M_a(p) + 1, & \text{if } p \in t^\bullet \setminus {}^\bullet t \\ M_a(p) - 1, & \text{if } p \in {}^\bullet t \setminus t^\bullet \\ M_a(p), & \text{otherwise} \end{cases}$$



(a) Transition t_1 is *enabled*

(b) Transition t_1 has been *fired*

Figure 2.1: An ordinary Petri net in two states illustrating the firing of a transition

Example 1: Illustration of an ordinary Petri net

Figures 2.1(a) and 2.1(b) depict an ordinary Petri net modelling two chemical reactions. The first chemical reaction, modelled by transition t_1, is the reaction between carbon (C) and dioxygen (O_2) which produces carbon dioxide (CO_2). The second chemical reaction, modelled by transition t_2, is the reaction between sodium hydroxide ($NaOH$) and carbon dioxide (CO_2) which produces sodium bicarbonate ($NaHCO_3$). The marking of figure 2.1(a) models a situation where one atom of carbon, one molecule of dioxygen, and one molecule of sodium hydroxide are present. In this marking the transition t_1 is *enabled* and *firing* it leads to the marking depicted by figure 2.1(b) which models a situation where one molecule of carbon dioxide and one molecule of sodium hydroxide are present, in this marking transition t_2 is *enabled*.

For clarity, modelling efficiency and convenience, ordinary Petri nets can be generalised by associating weight to arcs, leading to the notion of generalised Petri nets.

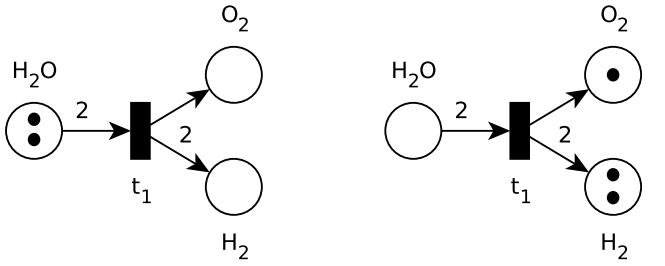
Definition 9: Generalised Petri net

A generalised Petri net N is a tuple $\langle P, T, F, W \rangle$ where:

- $\langle P, T, F \rangle$ is an ordinary Petri net, and
- $W : F \to \mathbb{N}$ is a function that assigns a weight to each arc.

In the context of a generalised Petri net a transition t is *enabled* in a marking M_a if and only if $\forall p \in {}^\bullet t, M_a(p) \geq W(p,t)$. When an *enabled* transition t is *fired*, it *consumes* $W(p,t)$ token(s) from each place p of ${}^\bullet t$ and *produces* $W(t,p)$ token(s) for each place p of $t^\bullet$. Formally, the firing of a transition t, enabled in a marking M_a, results in a new marking M_b defined as follows:

$$\forall\ p \in P,\ M_b(p) = \begin{cases} M_a(p) + W(t,p) - W(p,t), & \text{if } p \in t^\bullet \cap {}^\bullet t \\ M_a(p) + W(t,p), & \text{if } p \in t^\bullet \setminus {}^\bullet t \\ M_a(p) - W(p,t), & \text{if } p \in {}^\bullet t \setminus t^\bullet \\ M_a(p), & \text{otherwise} \end{cases}$$



(a) Transition t_1 is *enabled* (b) Transition t_1 has been *fired*

Figure 2.2: An generalised Petri net in two states illustrating the firing of a transition

Example 2: Illustration of a generalised Petri net

Figures 2.2(a) and 2.2(b) depict a generalised Petri net which models the electrolysis of water by the transition t_1. The marking of figure 2.2(a) models a situation where two molecules of water (H_2O) are present. In this marking the transition t_1 is *enabled* and *firing* it leads to the marking depicted by figure 2.2(b), which models a situation where one molecule of dioxigen (O_2) is present and two molecules dihydrogen (H_2) are present.

Note that generalised Petri nets and ordinary Petri nets have the same expressive power as a generalised Petri net can always be transformed into an ordinary Petri net, however the resulting ordinary Petri net is in general more complex than the generalised one.

Let $N = \langle P, T, F \rangle$ (resp. $N = \langle P, T, F, W \rangle$) be an ordinary Petri net (resp. a generalised Petri net). The Petri net N is said to be pure (i.a. self-loop-free) if and only if $\forall (x,y) \in (P \times T) \cup (T \times P), (x,y) \in F \Rightarrow (y,x) \notin F$.

Let $x \in P \cup T$, we denote $[x]$ the *cluster* of x defined as the smallest subset of $P \cup T$ such that $x \in [x]$, $p \in [x] \cap P \Rightarrow p^\bullet \subseteq [x]$ and $t \in [x] \cap T \Rightarrow t^\bullet \subseteq [x]$. The *set of clusters* of N is denoted as $C(N) = \{[x] \mid x \in P \cup T\}$.

Finally, it can be noted that Petri nets can also be viewed as *vector addition systems* [Leroux, 2011] as well as special kind of 2 – *automata* [Burroni, 1993].

2.1.2.1/ Matrix Representation of Petri nets

Ordinary as well as generalised Petri nets can be represented using their matrix form.

A matrix is a rectangular array of elements of a set. Let A be a set and $n, m \in \mathbb{N}$, we denote by $A^{(n,m)}$ the set of matrices of n rows and m columns of elements of the set A. Let $a \in A^{(n,m)}$, $i \in \{1, .., n\}$ and $j \in \{1, .., m\}$, we denote by $a_{i,j}$ the element of a at the i^{th} row and j^{th} column.

Let $N = \langle P, T, F \rangle$ (resp. $N = \langle P, T, F, W \rangle$) be an ordinary (resp. a generalised) Petri net such that $P = \{p_1, .., p_{|P|}\}$ and $T = \{t_1, .., t_{|T|}\}$.

In matrix form, a marking M of N is represented by a matrix $\vec{M} \in \mathbb{N}^{(|P|,1)}$ such that:

$$\forall i \in \{1, .., |P|\},\ \vec{M}_{i,1} = M(p_i)$$

A transition t is represented by a matrix $\vec{t} \in \mathbb{N}^{(|T|,1)}$ such that:

$$\forall i \in \{1, .., |T|\},\ \vec{t}_{i,1} = \begin{cases} 1, & \text{if } t = t_i \\ 0, & \text{otherwise} \end{cases}$$

The structure of N is represented by its pre-incidence matrix $\mathcal{W}^- \in \mathbb{N}^{(|P|,|T|)}$ and its post-incidence matrix $\mathcal{W}^+ \in \mathbb{N}^{(|P|,|T|)}$ such that:

$$\forall\ i \in \{1, .., |P|\},\ \forall\ j \in \{1, .., |T|\},\ \mathcal{W}^-_{i,j} = \begin{cases} 1 (resp.\ W(p_i, t_j)), & \text{if } (p_i, t_j) \in F \\ 0, & \text{otherwise} \end{cases}$$

$$\forall\ i \in \{1, .., |P|\},\ \forall\ j \in \{1, .., |T|\},\ \mathcal{W}^+_{i,j} = \begin{cases} 1 (resp.\ W(t_j, p_i)), & \text{if } (t_j, p_i) \in F \\ 0, & \text{otherwise} \end{cases}$$

In this representation, a transition $\vec{t}$ is *enabled* in a marking $\vec{M}_a$ if and only if:

$$\vec{M}_a \geq \mathcal{W}^- * \vec{t} \tag{2.1}$$

When an *enabled* transition $\vec{t}$ is *fired* in a marking $\vec{M}_a$ it leads to a new marking $\vec{M}_b$ such that:

$$\vec{M}_b = \vec{M}_a - \mathcal{W}^- * \vec{t} + \mathcal{W}^+ * \vec{t} \tag{2.2}$$

Assuming N is pure, equation 2.2 can be written as:

$$\vec{M}_b = \vec{M}_a + \mathcal{W} * \vec{t} \tag{2.3}$$

where $\mathcal{W} = \mathcal{W}^+ - \mathcal{W}^-$ is called the incidence matrix of N.

It follows that N can be fully described by the tuple $\langle \mathcal{W}^-, \mathcal{W}^+ \rangle$. Likewise, if N is pure then it can be fully described by the tuple $\langle \mathcal{W} \rangle$.

Example 3: Illustration of the matricial representation of a Petri net

Consider the generalised Petri net of Example 2 page 13 depicted in Figure 2.2(a) page 13. This generalised Petri net is composed of three places (H_2O, O_2 and H_2) and a transition (t_1). It can be fully described by the tuple $\langle \mathcal{W}^-, \mathcal{W}^+ \rangle$ where $\mathcal{W}^- = [-2, 0, 0]^\top$ and $\mathcal{W}^+ = [0, 1, 2]^\top$. Equivalently, as this generalised Petri net is pure, it can be fully described by its incidence matrix $\mathcal{W}$ defined as: $\mathcal{W} = [-2, 1, 2]^\top$. Let $\vec{M}_a = [2, 0, 0]^\top$ be the matricial representation of the marking of Figure 2.2(a) and $\vec{M}_b = [0, 1, 2]^\top$ be the matricial representation of the marking of Figure 2.2(b). We have $\vec{M}_b = \vec{M}_a + \mathcal{W} * \vec{t}$ where $\vec{t} = [1]$ is the matricial representation of transition t_1.

2.1.3/ Petri Nets Behavioural Properties

This section presents Petri nets behavioural properties in which we are interested.

In the context of Petri nets, one of the most important analysis problem, on which most verification problems are based, is called the *reachability problem*. The reachability problem is stated as follows: Given a Petri net N and M_0 an initial marking of N, can a given marking M of N be obtained from the marking M_0 after the firing of a sequence of transitions?

The conditions of a transition firing and its effects on the marking of a Petri net have been described in Section 2.1.2. Let M_a and M_b be two markings and t a transition of a Petri net N, we denote $M_a \xrightarrow{t} M_b$ the fact that transition t is *enabled* in marking M_a, and *firing* it results in the marking M_b. The marking M_b is denoted as *directly reachable* from M_a by transition t.

Let $M_1, M_2, .., M_n$ be markings and $\sigma = t_1, t_2, .., t_{n-1}$ a sequence of transitions of a Petri net N, we denote $M_1 \xrightarrow{\sigma} M_n$ the fact that $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} .. \xrightarrow{t_{n-1}} M_n$. The marking M_n is then said to be *reachable* from M_1 by the sequence of transitions σ. We denote $\mathcal{R}^N(M)$ the set of markings of N *reachable* from a marking M.

Definition 10: Reachability Problem

Given M a marking of a Petri net N whose initial marking is the marking M_0, the reachability problem is the problem of deciding whether $M \in \mathcal{R}^N(M_0)$.

Many relevant behavioural properties can be expressed and evaluated through the resolution of the reachability problem, notably safety properties. It has been shown that the *reachability problem* for Petri nets is decidable [Kosaraju, 1982, Leroux, 2011] but requires at least exponential space (and therefore time) to be solved in the general case [Lipton, 1976].

Another behavioural property of interest is called the *boundedness* property. An ordinary Petri net $N = \langle P, T, F \rangle$ (resp. a generalised Petri net $N = \langle P, T, F, W \rangle$) together with its initial marking M_0 is said *k-bounded* (or simply *bounded*) if and only if the number of tokens in each place does not exceed a finite number k for any markings reachable from M_0.

Definition 11: Boundedness

Let $N = \langle P, T, F \rangle$ (resp. $N = \langle P, T, F, W \rangle$) be an ordinary Petri net (resp. a generalised Petri net) whose initial marking is the marking M_0. The Petri net N is bounded if and only if $\exists\, k \in \mathbb{N},\ \forall\, M \in \mathcal{R}^N(M_0),\ \forall\, p \in P,\ M(p) \leq k$.

The last behavioural property considered is called the *liveness* property. With respect to the firing rule, a transition t is *dead* at marking M if it is not enabled in any marking M' reachable from M. A transition t is *live* if it is not dead in any marking reachable from the initial marking. A Petri net is said *live* if each of these transition is live.

Definition 12: Liveness

Let $N = \langle P, T, F \rangle$ (resp. $N = \langle P, T, F, W \rangle$) be an ordinary Petri net (resp. a generalised Petri net) whose initial marking is the marking M_0. The Petri net N is live if and only if $\forall\, t \in T,\ \forall\, M \in \mathcal{R}^N(M_0),\ \exists\, M_a, M_b \in \mathcal{R}^N(M),\ M_a \xrightarrow{t} M_b$.

2.1.4/ Petri Nets Subclasses

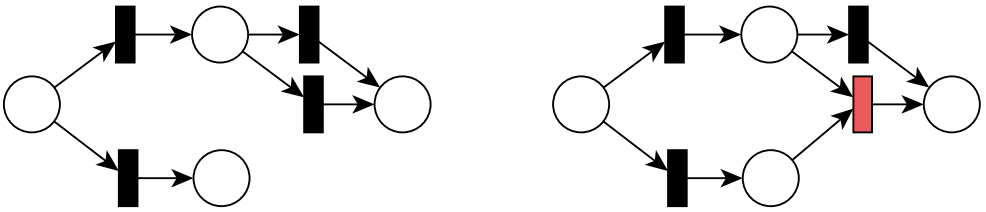
The need of considering classes of Petri nets sufficiently powerful with respect to their expressive capability has led to the definition of several subclasses of Petri nets. All such subclasses are defined only on the basis of structural characteristics. The four main Petri net subclasses are, in the order of growing expressiveness, defined as follow.

Definition 13: State Machine

Let $N = \langle P, T, F \rangle$ be an ordinary Petri net, N is a state machine (SM) if and only if:

$$\forall t \in T, |^\bullet(t)| = |(^\bullet t)| = 1.$$

A state machine is an ordinary Petri net without concurrency, but with possible conflicts among transitions. Figures 2.3(a) and 2.3(b) respectively illustrate a valid and an invalid state machine.



(a) A valid state machine (b) An invalid state machine

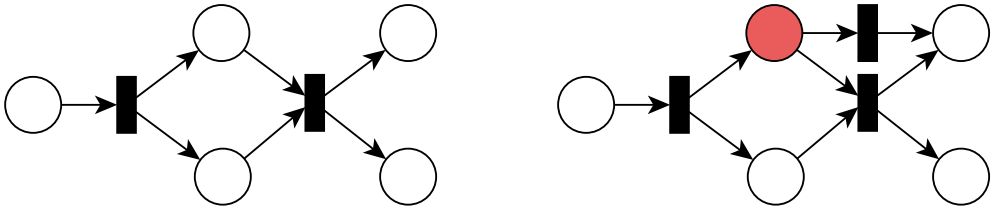
Figure 2.3: State machine

Definition 14: Marked Graph

Let $N = \langle P, T, F \rangle$ be an ordinary Petri net, N is a marked graph (MG) if and only if:

$$\forall p \in P, |^\bullet(p)| = |(^\bullet p)| = 1.$$

A marked graph is an ordinary Petri net without conflict, but there can be concurrency. Figures 2.4(a) and 2.4(b) respectively illustrate a valid and an invalid marked graph.



(a) A valid marked graph (b) An invalid marked graph

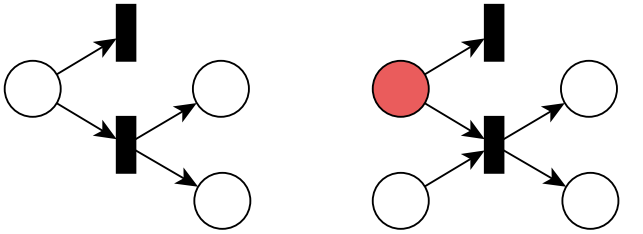
Figure 2.4: Marked graph

Definition 15: Free Choice Net

Let $N = \langle P, T, F \rangle$ be an ordinary Petri net, N is a free choice net (FC) if and only if:

$$\forall p \in P, (\mid p^\bullet \mid \leq 1) \vee ({}^\bullet(p^\bullet) = \{p\}).$$

A free choice net is an ordinary Petri net where there can be both concurrency and conflict, but not at the same time. Figures 2.5(a) and 2.5(b) respectively illustrate a valid and an invalid free choice net.



(a) A valid free choice net (b) An invalid free choice net

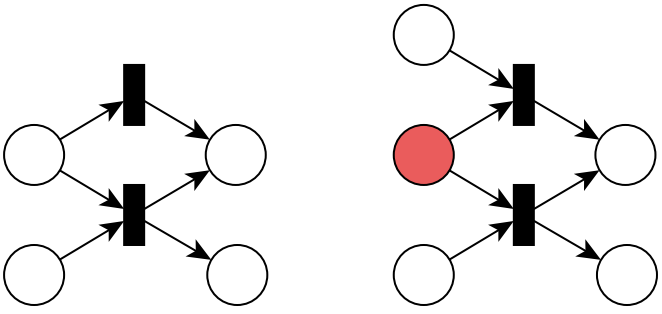
Figure 2.5: Free choice net

Definition 16: Asymmetric Choice Net

Let $N = \langle P, T, F \rangle$ be an ordinary Petri net, N is an asymmetric choice net (AC) if and only if:

$$\forall p_1, p_2 \in P, p_1^\bullet \cap p_2^\bullet = \emptyset \Rightarrow (p_1^\bullet \subseteq p_2^\bullet) \vee (p_1^\bullet \supseteq p_2^\bullet).$$

An asymmetric choice net is an ordinary Petri net where there can be both concurrency and conflict, but not symmetrically. Figures 2.6(a) and 2.6(b) respectively illustrate a valid and an invalid asymmetric choice net.



(a) A valid asymmetric choice net (b) An invalid asymmetric choice net

Figure 2.6: Asymmetric choice net

For a better visual interpretation of the respective complexity of these subclasses, Figure 2.7 summarises the inclusion relations among the presented Petri nets subclasses.

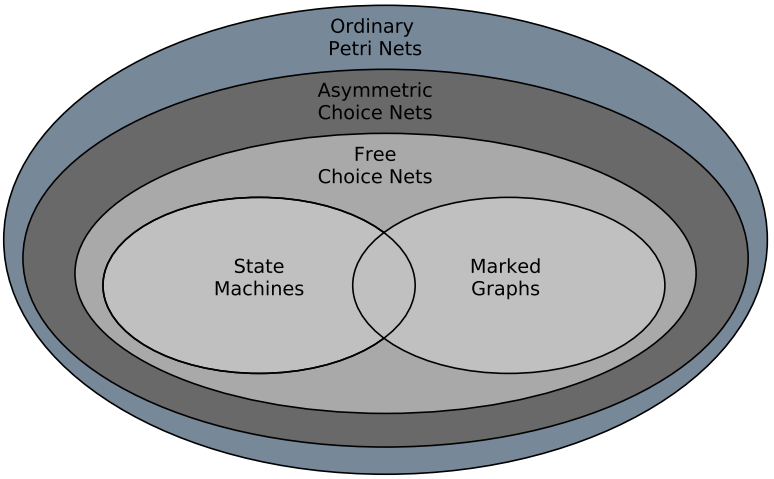


Figure 2.7: Venn diagram of Petri nets subclasses

2.1.5/ Petri Net Extensions

While Petri nets allow the modelling of most of real-life processes their specification ability and expressiveness can be improved. To this end, many extensions have been proposed. Some of them are completely backwards-compatible (i.e. are equivalent to Petri nets) and focus on improving the modelling capability of Petri nets while conserving the same expressiveness (e.g., Coloured Petri nets [Jensen, 1987] with finite colour's domains), whereas some others improve the expressive power of Petri nets (e.g., timed Petri nets [Ramchandani, 1974] which associate time with the firing of transitions).

The following sections present three Petri net extensions considered in this thesis: hierarchical Petri nets, coloured Petri nets and weighted transitions Petri nets.

2.1.5.1/ Hierarchical Petri Nets / Refinement

Modelling large and intricate Petri nets can be a difficult task and requires powerful structuring mechanisms [Dittrich, 1989]. Fortunately, similarly to modular programming, Petri nets can be designed using other Petri nets as building blocks [Fehling, 1991, Marechal et al., 2013, Marechal et al., 2015]. This modelling method enables modellers to substitute places and transitions by whole Petri nets (i.e. building blocks) while preserving properties of interest (e.g., liveness, boundedness).

To represent the models produced by such modelling method, hierarchical Petri net are used. Formally, a hierarchical Petri net is recursively defined as follows.

Definition 17: Hierarchical Petri Nets

The set $\mathcal{H}$ of hierarchical Petri nets is recursively defined as $H \in \mathcal{H} \Leftrightarrow H = \langle P_h, T_h, F_h \rangle$ where:

- $P_h \subseteq \mathcal{H}$ is a finite set of places,
- $T_h \subseteq \mathcal{H}$ is a finite set of transitions,
- $F_h \subseteq (P_h \times T_h) \cup (T_h \times P_h)$ is a set of arcs.

Hierarchical Petri nets are the support for an efficient modelling methodology called *stepwise refinement* [Suzuki et al., 1983]. Through the definition of a precise semantics of hierarchical Petri nets, stepwise refinement of Petri nets is a top-down approach to Petri nets development which consists in iteratively (i.e. step by step) refining a Petri net by substituting transitions and places by whole Petri nets until a sufficient level of detail is achieved. Each step of the refinement process transforms an abstraction of the modelled process (i.e. a Petri net with several details left-out) into a more precise abstraction of the modelled process. The inverse process of refining is called abstraction.

Petri nets obtained through stepwise refinement can be viewed as Petri nets with multiple layers of detail. In this way, stepwise refinement consists in explicitly giving a hierarchical abstraction of the modelled Petri net.

Note that, in the context of stepwise refinement of Petri nets, the semantic of hierarchical Petri nets is defined with respect to the construction of underlying Petri nets. Therefore, the use of hierarchical Petri nets focuses on improving the modelling capability of Petri nets while conserving the same expressiveness.

The main advantage of such a development is the possibility to derive properties of interest by construction. By imposing a precise semantics of hierarchical Petri nets as well as conditions on the building blocks used, each successive refinement can be proved to preserve properties of interest, such as liveness and boundedness [Suzuki et al., 1983].

2.1.5.2/ Coloured Petri Nets

Coloured Petri nets [Jensen, 1987] are an extension of Petri nets where data are assigned to the tokens and can be modified by transitions based on their contents. For data (also called colours) of finite domains they are equivalent to Petri nets with respect to their expressiveness. However they may have more succinct representations and may be more convenient for modelling.

Let Ξ be a non-empty set of *data-types* (also called colours), where each data-type is a set of *data-values*. We denote $\mathcal{L}(\mathcal{V}, \mathcal{W})$ the space of linear maps from $\mathcal{V}$ to $\mathcal{W}$, and O the zero map.

Definition 18: Coloured Petri net

A coloured Petri net (CPN) is a tuple $\langle P, T, C, W \rangle$ where:

- P is a finite set of places, T is a finite set of transitions, such that $P \cap T = \emptyset$,
- $C : P \cup T \to \Xi$ is the colour-function,
- $W^- : P \times T \to \mathcal{L}(\Xi, \Xi)$ is the pre-incidence function,
- $W^+ : P \times T \to \mathcal{L}(\Xi, \Xi)$ is the post-incidence function.

A *marking* of a coloured Petri net is a function M defined on P, such that $\forall\ p \in P, M(p) \in C(p) \rightarrow \mathbb{N}$. Two markings M_a and M_b are in relation $M_a \geq M_b$ if and only if $\forall p \in P, \forall\ c \in C(p), M_a(p)(c) \geq M_b(p)(c)$.

Let $\circledast$ denote the generalised matrix-multiplication where each product is replaced by a function composition. With this notation, a transition defined by $x(t) : C(t) \rightarrow \mathbb{N}$ (called a transition binding) is *enabled* in a marking M_a if and only if $M_a \geq W^-(t) \circledast x(t)$. When $x(t)$ is *enabled*, it may *fire*. If $x(t)$ *fires*, a new marking $M_b = M_a + (W^+ - W^-)(t) \circledast x(t)$ is reached. Similarly to Petri nets, the marking M_b is then said to be *directly reachable* from M_a by transition $x(t)$, written $M_a \xrightarrow{x(t)} M_b$, and the *reachability* relation is defined as the reflexive and transitive closure of the *direct reachability*.

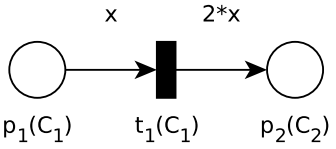


Figure 2.8: An example of coloured Petri net

Example 4: An example of coloured Petri net

Let $C_1 = \{1, 2, 3\}$ and $C_2 = \{2, 4, 6\}$ be two colors. The coloured Petri net depicted by Figure 2.8 is composed of two places (p_1 and p_2) and a transition (t_1) such that $C(p_1) = C(t_1) = C_1$ and $C(p_2) = C_2$. Transition t_1 has for effect to consume tokens of value x (1, 2 or 3) and produce token of value $2*x$ (2, 4 or 6). Its pre-incidence function is defined as $W^-(p_1, t_1) : C_1 \rightarrow C_1$ such that $W^-(p_1, t_1)([v_1, v_2, v_3]) = [v_1, v_2, v_3]$. Its post-incidence function is defined as $W^+(t_1, p_2) : C_1 \rightarrow C_2$ such that $W^+(t_1, p_2)([v_1, v_2, v_3]) = [v_1, v_2, v_3]$. Suppose an initial marking M_a such that $M_a(p_1) = [2, 0, 1]$ and $M_a(p_2) = [0, 0, 0]$, a marking where place p_1 contains two tokens of value 1 and one token of value 3, and place p_2 contains no token. The transition instance $(t, [1, 0, 1])$ is enabled in M_a, and firing it results in a marking M_b such that $M_a(p_1) = [1, 0, 0]$ and $M_a(p_2) = [1, 0, 1]$, a marking where place p_1 contains one token of value 1, and place p_2 contains one token of value 2 and one token of value 6.

Note that coloured Petri nets with finite colour domains and ordinary Petri nets have the same expressiveness. Indeed coloured Petri net with finite colour domains can always be transformed into an ordinary Petri net, however the resulting ordinary Petri net is in general more complex than the coloured one.

While Petri nets capture the essential process flows, they often get very large and complex when additional information is provided to refine the basic models. However, such refined models are often required to specify desired behaviours. In this context, using coloured Petri nets allows specifiers to incorporate data into tokens of the system model, e.g., characteristics to treated cases in the context of workflows, which enables them to represent complex behaviour in a concise manner.

2.1.5.3/ Weighted Transitions Petri Nets

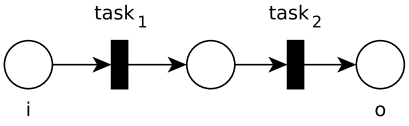
Weighted transitions Petri nets are an extension of Petri nets. They associate weights with transitions via a weight function. This enables performance analysis.

A weight function is defined within a framework where $(C, +_C)$ is a commutative monoid. We denote O_C the minimal element of C. Let $N = \langle P, T, F \rangle$ be an ordinary workflow net, a weight function of N is a total function $C : T \to C$ assigning a weight to each transition.

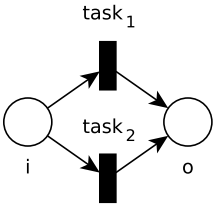
Let $N_w = \langle N, C \rangle$ be the weighted transitions Petri net formed from the ordinary Petri net N by associating a weight function C with it. The weight of a transition t is then given by $C(t)$. Let M_a, M_b be markings of N, and σ a transition sequence such that $M_a \xrightarrow{\sigma} M_b$, the weight associated with the execution σ is given by the execution weigh function $\mathfrak{C}$ defined by $\mathfrak{C}(\sigma) = \sum_{t \in T} O_t(\sigma) * C(t)$ where $O_t(\sigma)$ is the number of occurrences of t in σ

2.1.6/ Workflow Nets

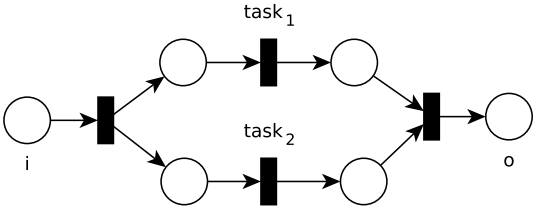
Workflow nets (WF-nets) are special cases of Petri net introduced by W.M.P. van der Aalst [van der Aalst, 1998]. They are used to model the control-flow dimension of a workflow. They allow the modelling of complex workflows exhibiting concurrencies, conflicts, as well as causal dependencies of activities (i.e tasks). The different activities are modelled by transitions, while causal dependencies are modelled by places and arcs. For instance, the Petri nets depicted in Figures 2.9(a), 2.9(b) and 2.9(c) are workflow nets respectively illustrating the structure of causal dependency, conflict and concurrency of two distinct tasks. Let us notice that, in the context of workflows, as places correspond to conditions, specifiers are used to considering ordinary Petri nets [van der Aalst, 1998].



(a) Causal dependency of $task_1$ and $task_2$



(b) Conflict of $task_1$ and $task_2$



(c) Concurrency of $task_1$ and $task_2$

Figure 2.9: Illustration of causal dependency, conflict and concurrency in workflow nets

Formally a workflow net is defined as follows.

Definition 19: Workflow net

An ordinary Petri net $N = \langle P, T, F \rangle$ is a Workflow net if and only if:

- N has two special places i and o, where ${}^\bullet i = \emptyset$ and $o^\bullet = \emptyset$, and
- for each node $n \in (P \cup T)$, there exists a path from i to o passing through n.

The places i and o respectively correspond to the beginning and termination of the processing of a case. Let $N = \langle P, T, F \rangle$ be a workflow net and $k \in \mathbb{N} \setminus \{0\}$ be a number of cases. The beginning of the processing of k cases corresponds to the initial marking $M_{i(k)}$ such that:

$$\forall\ p \in P,\ M_{i(k)}(p) = \begin{cases} k, & \text{if } p = i \\ 0, & \text{otherwise} \end{cases}$$

Analogously, the termination of the processing of k cases corresponds to the final marking $M_{o(k)}$ such that:

$$\forall\ p \in P,\ M_{o(k)}(p) = \begin{cases} k, & \text{if } p = o \\ 0, & \text{otherwise} \end{cases}$$

A sequence of transitions σ is an (partial) execution of N if there exist M_a, M_b two markings of N such that $M_a \xrightarrow{\sigma} M_b$. A *correct* execution of N is an execution σ such that $M_{i(k)} \xrightarrow{\sigma} M_{o(k)}$. The behaviour of N is defined as the set Σ_k of all its correct executions σ such that $M_{i(k)} \xrightarrow{\sigma} M_{o(k)}$. Given a transition t and an execution σ, the function $O_t(\sigma)$ is the number of occurrences of t in σ.

Example 5: Workflow net example: On-demand order delivery

To illustrate the use cases of workflow nets, let us consider the business process associated with an on-demand order delivery company such as a Pizza delivery company. Textually this process is described as follows. The process starts when receiving a call from a client. The client then proceeds to, in any order, state his order as well as some personal information (e.g., name, address). Once a client stated his order, the order is prepared then delivered. When presented with his order, a client can present one or several promotional coupons before paying his order using either a credit card or by cash. The process finishes upon receiving payment.
This process can be modelled by a workflow net as shown in Figure 2.10.

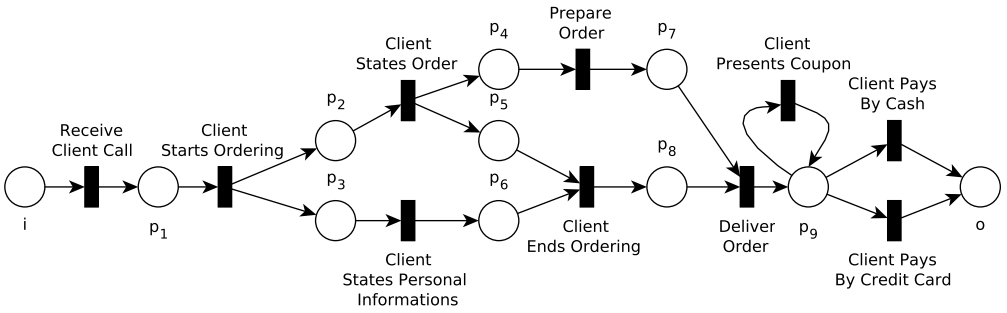


Figure 2.10: On-demand order delivery workflow net

2.1.7/ Workflow Nets Equivalences

Comparing the behaviour of workflow nets is a task which relies on the definition of equivalences. Let $N = \langle P, T, F \rangle$ be a workflow net and $k \in \mathbb{N} \setminus \{0\}$ be a number of cases. We present two kinds of equivalences. The first kind, called strong trace equivalence, compares workflow nets with respect to all of their transitions.

The behaviour of N for the processing of k cases, denoted Σ_k, is defined as the set of all its correct executions. Formally,

$$\Sigma_k = \{\sigma \mid M_{i(k)} \xrightarrow{\sigma} M_{o(k)}\}$$

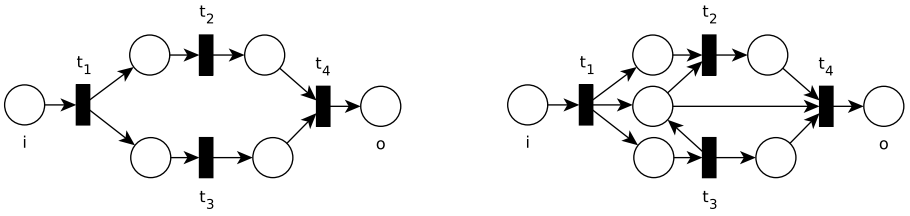
The strong trace set of N, denoted $\Gamma(N)$, is defined as:

$$\Gamma(N) = \cup_{k \in \mathbb{N} \setminus \{0\}} \Sigma_k$$

Two workflow nets are said to be strong trace equivalent if and only if their strong trace sets are equal. Formally,

Definition 20: Strong Trace Equivalence

Let $N_1 = \langle P_1, T_1, F_1 \rangle$ and $N_2 = \langle P_2, T_2, F_2 \rangle$ be two workflow nets.
N_1 and N_2 are strong trace equivalent if and only if $\Gamma(N_1) = \Gamma(N_2)$.



(a) (b)

Figure 2.11: Two strong trace equivalent workflow nets

Example 6: Illustration of strong trace equivalence

The two workflow nets respectively depicted by Figures 2.11(a) and 2.11(b) are strong trace equivalent. They have the same behaviour with respect to all of their transitions.

The second kind of equivalence, called weak trace equivalence, compares workflow nets with respect to a given subset of their transitions.

Let $S \subseteq T$ be the considered transition set. Transitions of S are said to be part of the visible behaviour of N whereas transitions of $T \setminus S$ are said to be part of the invisible behaviour of N. Let M_a and M_b two markings of N. We denote $M_a \Rightarrow_S M_b$ the fact that $M_a = M_b$ or there exists a sequence σ of transitions of $T \setminus S$ such that $M_a \xrightarrow{\sigma} M_b$. We write $M_a \overset{t}{\Rightarrow}_S M_b$ if and only if there exist M_1, M_2 two markings of N and $t \in S$ such that $M_a \Rightarrow_S M_1 \xrightarrow{t} M_2 \Rightarrow_S M_b$. For a sequence $\sigma = t_1, .., t_n$ of transitions of S, we write $M_a \overset{\sigma}{\Rightarrow}_S M_b$ if and only if there exist $n-1$ markings of N, denoted $M_1, .., M_{n-1}$, such that $M_a \overset{t_1}{\Rightarrow}_S M_1 \overset{t_2}{\Rightarrow} .. \overset{t_{n-1}}{\Longrightarrow} M_{n-1} \overset{t_n}{\Rightarrow}_S M_b$.

The behaviour of N for the processing of k cases with respect to the set of transitions S, denoted Σ_k^S, is defined as:

$$\Sigma_k^S = \{\sigma \mid M_{i(k)} \overset{\sigma}{\Rightarrow}_S M_{o(k)}$$

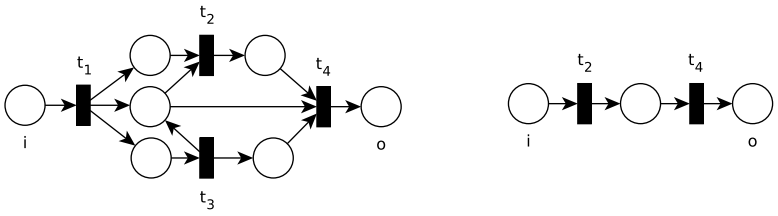
The weak trace set of N with respect to the set of transitions S, denoted $\Gamma_S(N)$, is defined as:

$$\Gamma_S(N) = \cup_{k \in \mathbb{N} \setminus \{0\}} \Sigma_k^S$$

Two workflow nets are said to be weak trace equivalent with respect to a set of transitions if and only if their weak trace sets with respect to this set of transitions are equal. Formally,

Definition 21: Weak Trace Equivalence

Let $N_1 = \langle P_1, T_1, F_1 \rangle$ and $N_2 = \langle P_2, T_2, F_2 \rangle$ be two workflow nets. Let S be a set of considered transitions such that $S \subseteq T_1$ and $S \subseteq T_2$.
N_1 and N_1 are weak trace equivalent with respect to S if and only if $\Gamma_S(N_1) = \Gamma_S(N_2)$.



(a) (b)

Figure 2.12: Two weak trace equivalent workflow nets with respect to $\{t_2, t_4\}$

Example 7: Illustration of weak trace equivalence

The two workflow nets respectively depicted by Figures 2.12(a) and 2.12(b) are weak trace equivalent with respect to $\{t_2, t_4\}$. They have the same behaviour with respect to the transitions t_2 and t_4.

2.1.8/ Workflow Nets Soundness

In the context of workflow nets, a well-established correctness feature that all workflows should verify is called *soundness* [van der Aalst, 1998]. It states that beside structural properties given by the definition of workflow nets, a *sound* workflow net describes a procedure that will terminate eventually (option to complete), and that when it does there is a token in place o and all of the other places are empty (proper completion). Additionally, a *sound* workflow net may not contain dead transitions. This correctness criteria constitute an underlying property of workflow nets that has to be verified in order to ensure correct executions.

Definition 22: Soundness [van der Aalst, 1998, van der Aalst et al., 2011]

Let $N = \langle P, T, F \rangle$ be a workflow net, N is sound if and only if:

- $\forall M \in \mathcal{R}^N(M_{i(1)}), M_{o(1)} \in \mathcal{R}^N(M)$ (option to complete),
- $\forall M \in \mathcal{R}^N(M_{i(1)}), (M(o) > 0) \Rightarrow (M = M_{o(1)})$ (proper completion), and
- $\forall t \in T, \exists M, M' \in \mathcal{R}^N(M_{i(1)}), M \xrightarrow{t} M'$ (no dead transitions).

Note that if we assume fairness – transitions that are enabled infinitely often will fire eventually – then the first requirement implies that eventually the final making is reached. As argued in [van der Aalst, 1998], the fairness assumption is reasonable in the context of workflow management. Indeed, all choices are made implicitly or explicitly by applications, humans or external actors which should not introduce infinite loops.

To apply the notion of soundness to workflow nets handling multiple cases at once, the notion of k-soundness introduced in [Barkaoui et al., 2007] (also known as structural soundness) extends the classical soundness to the processing of k cases and is proved to be decidable [Ţiplea et al., 2005].

Definition 23: k-soundness [Barkaoui et al., 2007]

Let $N = \langle P, T, F \rangle$ be a workflow net, and $k \in \mathbb{N}$. N is k-sound if and only if:

- $\forall M \in \mathcal{R}^N(M_{i(k)}), M_{o(k)} \in \mathcal{R}^N(M)$
- $\forall t \in T, \exists M, M' \in \mathcal{R}^N(M_{i(k)}), M \xrightarrow{t} M'$

Furthermore, a workflow net is said *generalised sound* if it is *k-sound* for all $k \in \mathbb{N}$. Generalised soundness is also proved to be decidable [Van Hee et al., 2004].

Definition 24: Generalised Soundness [Barkaoui et al., 2007]

Let $N = \langle P, T, F \rangle$ be a workflow net, N is generalised sound if and only if:

$$\forall k \in \mathbb{N}, N \text{ is } k\text{-sound}$$

2.1.9/ Stepwise Refinement of Worflow Nets

In Section 2.1.5.1, we defined hierarchical Petri nets and their relation to stepwise refinement, a powerful development method which consists in iteratively substituting places and transitions of a Petri net by whole Petri nets (i.e. building blocks) while preserving properties of interest (e.g., liveness, boundedness) until the desired level of detail is achieved. Such construction is often native to workflow modelling standard. For instance, in BPMN [Allweyer, 2016], processes can contain sub-processes which are themselves processes. This allows modellers to start with a simple model of the considered process (i.e. a rough abstraction of the process) composed of abstract activities. Such activities are then iteratively detailed by the processes which model them until a sufficient level of detail is reached. Doing so enables modellers to get a general idea of the process, recognize correlations, and identify where the weak points are early, concerns that are essential to reliable and effective process modelling.

In the context of workflow nets, this leads to a simple and intuitive stepwise refinement method based on the substitution of transitions and places by whole workflow nets which preserves generalised soundness [Van Hee et al., 2003].

A composed workflow net built using this method has special transitions/places that represent several whole (composed or not) workflow nets. Such composed workflow nets can then be viewed as hierarchical Petri nets, called hierarchical workflow nets and representing workflow nets with multiple layers of detail.

Intuitively, a single task (i.e. a single transition) on a higher level can become a sequence of subtasks (i.e. a sequence of transitions) involving choices and concurrencies. Likewise, a resource (i.e. a token) meeting a condition (i.e. at some place) can require a sequence of subtasks (i.e. a sequence of transitions) involving choices and concurrencies to be executed between the moment they meet the condition and the moment they are available by other tasks.

Formally, a hierarchical workflow net is recursively defined as follow.

Definition 25: Hierarchical Workflow Nets

The set $\mathcal{W}$ of hierarchical workflow nets is recursively defined as $W \in \mathcal{W} \Leftrightarrow W = \langle P_h, T_h, F_h \rangle$ where:

- $P_h \subseteq \mathcal{W}$ is a finite set of places,
- $T_h \subseteq \mathcal{W}$ is a finite set of transitions,
- $F_h \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs,
- the Petri net $\langle P_h, T_h, F_h \rangle$ is a workflow net or $P_h = T_h = F_h = \emptyset$

The semantics of a hierarchical workflow net W is defined with respect to its underlying Petri net which assumes standard Petri nets semantics. Formally, let $N_1 = \langle P_1, T_1, F_1 \rangle$ and $N_2 = \langle P_2, T_2, F_2 \rangle$ be two workflow nets, there exist two basic refinement operations.

The first refinement operation is the *place refinement* of a place $p \in P_1$ with a workflow net N_2 which produces a new workflow net $N = N_1 \otimes_p N_2$ built by replacing the place p by the workflow net N_2: p is removed from N_1, the input transitions of p (i.e. ${}^\bullet p$) become the input transitions of the initial place of N_2 and the output transitions of p (i.e. $p^\bullet$) become the output transitions of the final place of N_2. This operation is illustrated by Figure 2.13.

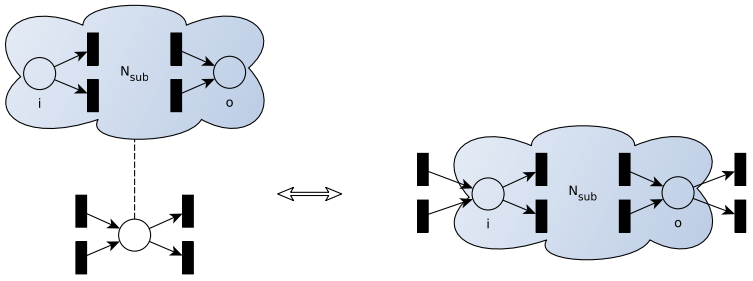


Figure 2.13: Illustration of the place refinement of place p with a workflow net N_{sub}

The second refinement operation is the *transition refinement* of a transition $t \in T_1$ with a workflow net N_2 which produces a new workflow net $N = N_1 \otimes_t N_2$ built by replacing the transition t by the workflow net N_2: t is removed from N_1, the initial place $i_2 \in N_2$ and the final place $o_2 \in N_2$ are removed from N_2, the input places of p (i.e. ${}^\bullet p$) become the input places of the output transition of i_2 (i.e. $i_2^\bullet$) the initial place of N_2, the output places of p (i.e. $p^\bullet$) become the output places of the input transition of o_2 (i.e. ${}^\bullet o_2$) the final place of N_2. This operation is illustrated by Figure 2.14.

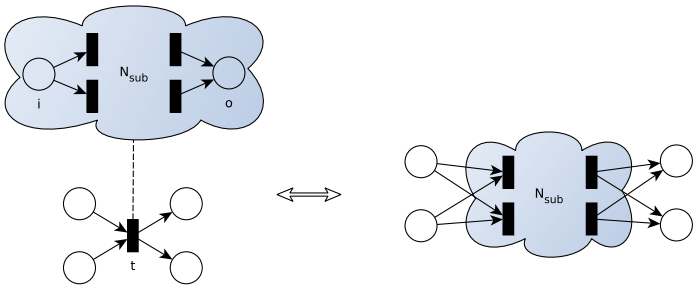


Figure 2.14: Illustration of the transition refinement of transition t with a workflow net N_{sub}

It follows that the underlying w net of a hierarchical workflow net can recursively be constructed as depicted by the function *underlying* of Algorithm 1.

Data: $W = \langle P_h, T_h, F_h \rangle$ a hierarchical workflow net
Result: $N = \langle P, T, F \rangle$ the workflow net underlying W
Function underlying($W = \langle P_h, T_h, F_h \rangle$)
  $N = \langle P_h, T_h, F_h \rangle$;
  forall the $S = \langle P_s, T_s, F_s \rangle \in P_h \cup T_h$ **do**
    if $P_s \neq \emptyset$ **then**
      $N = N \otimes_S underlying(S)$;
    end
  end
  return N

Algorithm 1: Construction of the workflow net underlying a hierarchical workflow net

By analogy with spacial dimensions, the presented refinement operations – place refinement and transition refinement – are called vertical refinement operations. This is because, given $W = \langle P_h, T_h, F_h \rangle$ a hierarchical workflow net, refinement operations, called horizontal refinement operations, can modify its flow relation (i.e. F_h) or add/remove nodes without modifying any of its nodes (i.e. the hierarchical workflow nets in the set $P_h \cup T_h$) and super-nodes (i.e. the hierarchical workflow nets whose set of nodes contains W). In the context of Petri nets, horizontal refinement operations are also called synthesis rules. Several synthesis rules are presented in Section 2.2.3.2 page 36, as well as in the contribution of this thesis in Section 4.1 page 68.

As previously said, a major advantage of stepwise refinement development method is its ability to preserve properties of interest. For example, the previously presented refinement operations – place refinement and transition refinement – are known to preserve generalised soundness [Van Hee et al., 2003].

Formally, let $N_1 = \langle P_1, T_1, F_1 \rangle$ and $N_2 = \langle P_2, T_2, F_2 \rangle$ be two generalised sound workflow nets, then $\forall\, n \in P_1 \cup T_1, N_1 \otimes_n N_2$ is a generalised sound workflow net.

While hierarchical workflow nets do not add any expressiveness to workflow nets, they greatly simplify the modelling work by allowing to model small parts of the whole process described as (hierarchical) workflow nets that are then composed into a hierarchical workflow net. It also fosters the re-usability of (hierarchical) workflow nets, a key notion which allows rapid development of new hierarchical workflow nets based on a sound library of generic (hierarchical) workflow nets developed previously.

Note that, in the context of stepwise refinement using place refinements and transition refinements,

the hierarchical workflow nets are explicitly given by the modellers. In a broader context, hierarchical workflow nets can be automatically and efficiently constructed from workflow nets through their decomposition [Vanhatalo et al., 2007, Polyvyanyy et al., 2011, Koehler et al., 2014].

2.1.10/ Modal Specification

To help engineers in their specification and validation activities, modal specifications [Larsen, 1989] have been designed to allow *loose* specifications of models under development. Modal specifications impose restrictions on the possible refinements by indicating whether activities are *necessary* or *admissible*. Those specifications are notably used within refinement approaches for software development. Modalities allow underspecification of certain activities which must be or may be present in later refinements. They provide a flexible tool for workflow development as decisions can be delayed to later steps of the development life cycle, when performing workflow refinements (Section 2.1.9).

In the framework of Petri nets, modal specifications allow specifiers to indicate that a transition (i.e. an activity) is necessary or just admissible. This concept provides two kinds of transitions: the *must*-transitions and the *may*-transitions [Elhog-Benzina et al., 2012]. More precisely, in the context of workflow nets, a *may*-transition (resp. *must*-transition) is a transition fired by at least one execution (resp. all the executions) of the process modelled by a workflow net.

Definition 26: Modal Workflow net

A modal workflow net M is a tuple $\langle N, T_\Box \rangle$ where:

- $N = \langle P, T, F \rangle$ is a workflow net, and
- $T_\Box \subset T$ is a set of *must*-transitions.

The set of *may*-transitions is the set of transitions T of N.

Formally, let $M = \langle \langle P, T, F \rangle, T_\Box \rangle$ be a modal workflow net, we have:

$$t \in T \Leftrightarrow \exists\, \sigma \in \Sigma_1, O_t(\sigma) > 0, \text{ and } t \in T_\Box \Leftrightarrow \forall\, \sigma \in \Sigma_1, O_t(\sigma) > 0$$

Example 8: Illustration of a modal specification

For example, consider the on-demand order delivery workflow net described in Figure 2.10 page 22. Some activities of this workflow are *necessary* (e.g., *Client States Order*, *Deliver Order*) whereas some activities of this workflow are *admissible* (e.g., *Client Presents Coupon*). More precisely, an example of a modal specification of this workflow is given by the definition of the set of *must*-transitions containing the following transitions: *Receive Client Call*, *Client States Order*, *Client States Personal Informations*, *Prepare Order*, *Deliver Order*.

2.2/ Analysis of Petri Nets

Beside being a computational modelling suited to the descriptions of complex systems exhibiting causal dependencies, conflicts and concurrencies, Petri nets rely on a strong theoretical and formal background allowing modellers to draw important conclusions about the modelled systems without

having to resort to the time and cost ineffective trial and error prototyping. Thereby modellers can answer questions about what the modelled system behaviour is supposed to be under specific operational conditions, what properties are inherent to the structure of the net, what to expect and what not to expect from the system during operation.

When it comes to analysing Petri nets, there exist three major approaches:

- Behavioural approach - based on the construction of reachability (coverability) graphs,
- Structural approach - based on the matrix representation of Petri nets and algebraic techniques as well as on topological features, and
- Reduction (abstraction) approach - based on transformations simplifying Petri nets (e.g., reducing the size of Petri nets)

On the one hand, the first approach essentially relies on the enumeration of all reachable markings. It can by applied to arbitrary Petri nets but is limited to small instances of Petri nets due the complexity arising from the state-space explosion. On the other hand, the two other approaches are not as generic but have been shown to be very powerful over specific subclasses of Petri nets or specific cases.

The following sections briefly describe these three approaches.

2.2.1/ Behavioural Approach

As previously said, the behavioural approach essentially relies on the enumeration of all markings reachable from an initial state. The result of such an enumeration is stored in a labelled graph where nodes represent markings, and each arc represents a transition firing which transforms a marking into another.

Let $N = \langle P, T, F \rangle$ be an ordinary Petri net, and $M_0 : P \to \mathbb{N}$ be an initial marking of N. The reachability graph $RG(N, M_0)$ of the ordinary Petri N with initial marking M_0 is a rooted, directed graph $\langle V, E, v_0 \rangle$ where:

- $V = \mathcal{R}^N(M_0)$ is the set of nodes (i.e. each marking of N reachable from the initial marking M_O is a node),
- $v_0 = M_0$ is the root node (i.e. the initial marking M_O of N is the root node), and
- $E = \{\langle M, t, M' \rangle \, | M, M' \in V, t \in T, M \xrightarrow{t} M'\}$ is the set of arcs (i.e. the firing of transitions between markings of N reachable from the initial marking M_O).

Reachability graphs aim at representing the underlying semantics of Petri nets from which analysis can be carried out on. However, such representation may be infinite whenever the considered Petri net is unbounded.

To overcome this issue, one can represent the reachability graph by mean of abstraction. A well-known abstract reachability graph is the coverability graph [Karp et al., 1969]. The main idea of the coverability graph is to represent infinite parts of the reachability graph where some places become unbounded in a finite number of nodes thus leading to finite over-approximation of the reachability graph.

While many properties can be checked using the coverability graph (e.g., boundedness, dead transition) others may not in general be checked due to the the fact that it defines an over-approximation (e.g., reachability, liveness).

In particular, the coverability graph can be used to decide whether a workflow net is sound as well as modal specifications validity.

The reachability graphs and the coverability graphs of Petri nets (resp. workflow nets) provide the basis of a verification technique called model checking. Model checking is a verification technique that explores all possible states of a system in a brute-force manner [Clarke et al., 1999]. It aims at verifying the validity of a desired behavioural property (the specification), usually expressed in temporal logic such as Linear Temporal Logic [Gabbay et al., 1980] or Computation Tree Logic [Clarke et al., 1981], via an exhaustive explicit or symbolic enumeration of all of the reachable states and transitions between them. If the property is found not to hold in all system executions, a counterexample is produced, consisting of a trace of the model from a start state to an error state in which the specification is violated, providing a very helpful approach for debugging the system design. Examples of Petri nets model checkers include LoLA [Schmidt, 2000] and TINA [Berthomieu* et al., 2004].

However, a main challenge of model checking is the state explosion problem [Valmari, 1998]. Indeed, the complexity of the algorithm constructing the reachability graph as well as the coverability graph can be worse than primitive recursive space. This renders model checking intractable on large instances of Petri nets (resp. workflow nets).

2.2.2/ Structural Approach

This section presents the structural approach used to analyse Petri nets. Structural analysis techniques are mainly based on the static structure of a Petri net, and aim at deriving links between the topological structure of a Petri net and its behaviours. They mainly revolve around algebraic techniques applied to the matrix representation of Petri nets.

The following sections describe the main features linked to the structural analysis of Petri nets.

2.2.2.1/ State Equation

From the matrix representation of Petri nets (Section 2.1.2.1 page 14), one would like to give an algebraic description of how the markings of Petri nets change (evolve).

Let $N = \langle P, T, F \rangle$ (resp. $N = \langle P, T, F, W \rangle$) be an ordinary (resp. a generalised) Petri net such that $P = \{p_1, .., p_{|P|}\}$ and $T = \{t_1, .., t_{|T|}\}$.

Assuming, without loss of generality, that N is pure, then N is fully described by its incidence matrix $\mathcal{W}$.

Let M_a, M_b be two markings of N and t a transition of N. By definition of the firing rule we have:

$$M_a \xrightarrow{t} M_b \Leftrightarrow \vec{M_b} = \vec{M_a} + \mathcal{W} * \vec{t} \geq 0 \tag{2.4}$$

The right hand side of the equivalence in equation 2.4 is a state equation: it relates an *actual state* $\vec{M_a}$, to a *next state* $\vec{M_b}$ obtained after firing transition $\vec{t}$.

Next, this equation is extended to consider transition sequences.

Let σ be a transition sequence of N, in matrix form, σ is represented as a matrix $\vec{\sigma} \in \mathbb{N}^{(|T|,1)}$ such that $\forall i \in \{1, .., |T|\},\ \vec{\sigma}_{i,1} = O_{t_i}(\sigma)$.

Building up on equation 2.4 by using the distributivity property of the multiplication operator ($*$) we obtain the state equation:

$$M_a \xrightarrow{\sigma} M_b \Rightarrow \vec{M}_b = \vec{M}_a + \mathcal{W} * \vec{\sigma} \geq 0 \tag{2.5}$$

Similarly to the right hand side of the equivalence in equation 2.4, the right hand side of the equivalence in equation 2.5 is a state equation: it relates an *actual state* $\vec{M}_a$, to a *next state* $\vec{M}_b$ obtained after firing transition sequence $\vec{\sigma}$.

Equation 2.5 is called the state equation of N, also known as the fundamental equation of N. Intuitively, the state equation considers, based on the structure of N, the transition sequences such that for each place of N the number of tokens produced added to the tokens present in the starting marking of the place is equal to the number of tokens consumed added to the tokens present in the reached marking of the place. Note that the order in which the transitions involved in the transition sequences are fired is not considered by the state equation.

It provides a necessary condition to the existence of a transition sequence σ starting from marking M_a and leading to marking M_b. However, it is important to note that it does not provide a sufficient condition to the existence of a transition sequence σ starting from marking M_a and leading to marking M_b. More precisely, a non-negative integer solution of the equation $\vec{M}_b = \vec{M}_a + \mathcal{W} * \vec{\sigma}$ does not imply that a transition sequence σ such that $M_a \xrightarrow{\sigma} M_b$ exists.

Solutions satisfying the state equation of N that do not correspond to valid executions of N are called spurious solutions. The existence of spurious solutions is one of the main problems in the utilization of algebraic linear techniques to analyse the behaviour of a Petri net.

Let us note that the state equation is the basis of a reachability analysis approach based on the concept of counterexample guided abstraction refinement (CEGAR) [Clarke et al., 2000]. In essence, [Wimmel et al., 2011] iteratively analyses spurious solutions of the state equation and add constraints that exclude a solution found to be spurious but do not exclude any consistent solutions.

2.2.2.2/ Structural Invariants

This section describes the different structural invariants that arise from the study of the incidence matrix of Petri nets. Structural invariants are structural features independent of the initial marking of a Petri net from which behavioural properties can be inferred.

Let $N = \langle P, T, F \rangle$ (resp. $N = \langle P, T, F, W \rangle$) be an ordinary (resp. a generalised) Petri net such that $P = \{p_1, .., p_{|P|}\}$ and $T = \{t_1, .., t_{|T|}\}$. Assuming, without loss of generality, that N is pure, then N is fully described by its incidence matrix $\mathcal{W}$. There exist two main categories of invariants: T-invariants and P-invariant.

Definition 27: T-invariant

Let $\vec{I} \in \mathbb{Z}^{(|T|,1)}$ be a matrix such that $\vec{I} \neq \vec{0}$, $\vec{I}$ is a T-invariant of N if and only if:

$$\mathcal{W} * \vec{I} = \vec{0} \tag{2.6}$$

A T-invariant $\vec{I}$ is called a T-semiflow if and only if:

$$\forall i \in \{1, .., |T|\},\ \vec{I}_{i,1} \geq 0 \tag{2.7}$$

The support of a T-invariant $\vec{I}$, noted $||\vec{I}||$, is defined as:

$$||\vec{I}|| = \{\ t_i \mid \vec{I}_{i,1} > 0\ \} \tag{2.8}$$

Intuitively, a T-invariant of N indicates a possible loop in the net, i.e. a sequence of transitions which does not change the marking of N.

Theorem 1: T-invariant Property

Let M_0 be the initial marking of N, M be a marking of N and σ a transition sequence such that $M_0 \xrightarrow{\sigma} M$. $M = M_0$ if and only if $\vec{\sigma}$ is a T-invariant of N.

The Petri net N is said to be covered by T-invariants if and only if, for each transition $t \in T$, there exists $\vec{I}$, a T-invariant of N, such that $t \in ||\vec{I}||$. The boundedness and liveness of N can be inferred from T-invariants as follows.

Theorem 2: T-invariants Covered Petri Net Property

If N is covered by T-invariants then N is bounded and live.

Definition 28: P-invariant

Let $\vec{I} \in \mathbb{Z}^{(1,|P|)}$ be a matrix such that $\vec{I} \neq \vec{0}$, $\vec{I}$ is a P-invariant of N if and only if:

$$\vec{I} * \mathcal{W} = \vec{0} \tag{2.9}$$

A P-invariant $\vec{I}$ is called a P-semiflow if and only if:

$$\forall i \in \{1, .., |P|\},\ \vec{t}_{1,p} \geq 0 \tag{2.10}$$

The support of a P-invariant $\vec{I}$, noted $||\vec{I}||$, is defined as:

$$||\vec{I}|| = \{\ p_i \mid \vec{I}_{1,i} > 0\ \} \tag{2.11}$$

Intuitively, a P-invariant corresponds to a set of places whose weighted token count is a constant for any reachable marking.

Theorem 3: P-invariant Property

Let M_0 be the initial marking of N and $M \in \mathcal{R}^N_{M_0}$ be a marking of N. If $\vec{I}$ is a P-invariant of N then:

$$\vec{I} * M_0 = \vec{I} * M \tag{2.12}$$

This way, P-invariants are notably used to prove mutual exclusion of places.

The Petri net N is said to be covered by P-invariants if and only if, for each place $p \in P$, there exists $\vec{I}$, a P-invariant of N, such that $p \in ||\vec{I}||$. The boundedness of N can be inferred from P-invariants as follows.

Theorem 4: P-invariant Covered Petri Net Property

If N is covered by P-invariants then N is bounded.

Let us note that any linear combination of P-invariants (resp. T-invariants) is a P-invariant (resp. T-invariant).

2.2.2.3/ Siphons and Traps

Siphons and traps are interesting features that can, as invariants, be used to infer behavioural properties of a Petri net.

Let $N = \langle P, T, F \rangle$ (resp. $N = \langle P, T, F, W \rangle$) be an ordinary (resp. a generalised) Petri net.

Definition 29: Siphon

Let $N \subseteq P$ such that $N \neq \emptyset$:

- N is a siphon if and only if ${}^\bullet N \subseteq N^\bullet$.

Intuitively, a siphon is a subset of places such that every input transition of a place is an output transition of some other places.

Lemma 1: Siphon property[Murata, 1989]

An unmarked siphon cannot be marked.

Definition 30: Trap

Let $N \subseteq P$ such that $N \neq \emptyset$:

- N is a trap if and only if $N^\bullet \subseteq {}^\bullet N$.

Intuitively, a trap is a subset of places such that every output transition of a place is an output transition of some other places.

Lemma 2: Trap property[Murata, 1989]

A marked trap cannot be unmarked.

We note that in the case of ordinary Petri nets, the presence of P-semiflows is linked with the presence of traps and siphons.

Lemma 3: P-Semiflow Siphon/Trap link property[Li et al., 2009]

Let $N = \langle P, T, F \rangle$ be an ordinary Petri net. A function $\omega : P \to \mathbb{N}$ is a P-semiflow of N if and only if $||\omega||$ the support of ω (i.e. $||\omega|| = \{p \in P | \omega(p) > 0\}$) is a trap and a siphon.

In the case of free-choice nets, there is a well-known relation between traps/siphons and liveness based on the evaluation of the so called *siphon-trap property*.

Definition 31: Siphon-Trap Property

Let N be a Petri net, N is said to satisfy the *siphon-trap property* (STP) if and only if:

- every siphon of N contains a marked trap.

Theorem 5: Free-Choice Nets liveness[Murata, 1989]

Let N be a free-choice Petri net. N is live if and only if N satisfies the *siphon-trap property*.

In the case of arbitrary Petri nets, the Siphon-Trap property can be linked to behavioural property as follows.

Theorem 6: Siphon-Trap conclusions[Murata, 1989]

Let N be a Petri net.

- If N is live then N satisfies the *siphon-trap property*.
- If N satisfies the *siphon-trap property* then N does not contain dead locks (i.e. all reachable markings enable at least one transition).

2.2.2.4/ Analysis of Workflow Nets

In this section we focus on the structural analysis techniques used to verify the *soundness* (resp. *k-soundness*, *generalised soundness*) property of workflow nets as presented in [Van der Aalst, 1997, Barkaoui et al., 2007, Van Hee et al., 2004].

Given a workflow net $N = \langle P, T, F \rangle$, the goal is to structurally decide whether N is k-sound.

To this end, the k-closure of a workflow net N, noted N_k^*, is defined as follows.

Definition 32: k-closure of a Workflow Net

Let $N = \langle P, T, F \rangle$ be a workflow net. The k-closure of N, noted N_k^* is defined as a generalised Petri net $\langle P^*, T^*, F^*, W^* \rangle$ where:

- $P^* = P$,
- $T^* = T \cup \{t^*\}$,
- $F^* = F \cup \{(o, t^*), (t^*, i)\}$,
- $\forall\ f \in F, W^*(f) = 1$, and
- $W^*(o, t^*) = W^*(t^*, i) = k$.

It is proved that the *k-soundness* property of a workflow net can be reduced to the liveness and boundedness of the corresponding k-closure Petri net.

Theorem 7: *k-soundness* via k-closure [Barkaoui et al., 2007]

A workflow net N is k-sound if and only if its k-closure Petri net N_k^* is live and bounded.

This allows structural analysis to be applied to the determination of the *k-soundness* property of a workflow net.

In the case of free-choice workflow nets, this leads to a polynomial time algorithm capable of deciding *k-soundness* [Barkaoui et al., 2007] based on the following theorem.

Theorem 8: Free-choice *k-soundness* [Barkaoui et al., 2007]

Let N be a workflow net such that its k-closure Petri net N_k^* is a free-choice Petri net. The workflow net N is k-sound if and only if N_k^* is covered by P-semiflows and satisfies the *siphon-trap* property.

However, in the general case this analysis only defines sufficient conditions for *k-soundness* based on the following theorem.

Theorem 9: Sufficient conditions for *k-soundness* [Van der Aalst, 1997]

Let N by a workflow net and N_k^* its k-closure Petri net. If the rank of the incidence matrix of N_k^* is equal to the number of clusters of N_k^* minus 1, and every proper siphon of N_k^* contains at least a token, then the workflow net is k-sound.

For some subclasses of workflow nets (e.g., free choice workflow nets), it has been shown that classical soundness, i.e. (weak) 1-soundness, implies generalised soundness [Ping et al., 2004]. In the context of arbitrary workflow nets, generalised soundness has been proved decidable and a decision procedure has been given in [Van Hee et al., 2004]. This procedure relies on the fact that a workflow net N is generalized sound if and only if a certain *batch workflow net* N' can be derived from it, and N' is generalized sound. The derivation is straightforward and only uses structural analysis of the net. Further, verifying the generalized soundness of N' is reduced to a finite number of proper termination checks in N'. We also note that in [van Hee et al., 2006b] the procedure described in [Van Hee et al., 2004] is improved by reducing the size of needed proper termination checks.

2.2.3/ Reduction Approach

This section presents an approach based on successive application of Petri net transformation rules preserving properties of interest (e.g., liveness, boundedness).

The main idea behind this approach is to iteratively apply a set of Petri net transformation rules (called a *kit*), each one reducing the size of the analysed Petri net while preserving the subset of properties to be analysed until the reduced Petri net is irreducible with respect to the considered set of transformation rules (i.e. until a fixed point is reached). The obtained irreducible Petri nets can be so simple that the properties under study can be trivially checked. Otherwise, further analysis are required to conclude.

The main advantage of this method is its ability to reduce the size of Petri nets under analysis while preserving a subset of their properties (e.g., liveness, boundedness). It follows that this approach is mainly considered as a pre-processing step to standard analysis approaches such as the one presented in Sections 2.2.1 page 29 and 2.2.2 page 30 enabling them to be carried out on instances smaller than the original ones.

2.2.3.1/ Formalization

Formally, a Petri net transformation rule ϕ is defined as a binary relation over Petri nets. It is fully described by the *conditions of application* under which it can be applied to a *source* Petri net, and the *construction algorithm* that is applied to the *source* Petri net to form a *target* Petri net.

Let N, $\tilde{N}$ be two Petri nets and ϕ a transformation rule, the fact that ϕ is *applicable* to N and that applying ϕ to N results in $\tilde{N}$, is denoted $(N, \tilde{N}) \in \phi$.

A Petri net transformation rule ϕ is called a Petri net reduction rule if for all $(N, \tilde{N}) \in \phi$, the number of nodes (i.e. the number of places and transitions) of $\tilde{N}$ is strictly smaller than the number of nodes of N. Conversely, a Petri net transformation rule ϕ is called a Petri net synthesis rule if for all $(N, \tilde{N}) \in \phi$, the number of nodes of $\tilde{N}$ is strictly greater than the number of nodes of N.

Note that refinement operations such as place refinement and transition refinement (Section 2.1.9 page 25) are synthesis rules. Likewise, reduction rules are abstraction operations.

Let ψ be a Petri net property such as liveness or boundedness, we denote $N \models \psi$ the fact that the Petri net N satisfies ψ.

If for all Petri nets N and $\tilde{N}$ such that $(N, \tilde{N}) \in \phi, N \models \psi \Rightarrow \tilde{N} \models \psi$, ϕ is said to *preserve* ψ. If the reverse holds as well (i.e. $\tilde{N} \models \psi \Rightarrow N \models \psi$), then ϕ is said to *strongly preserve* ψ.

The relation over Petri nets induced by a set of Petri net transformation rules is called a *kit*. Let Φ be the *kit* induced by n Petri net transformation rules $\phi_1, .., \phi_n$, Φ defines a binary relation over Petri nets such that $(N, \tilde{N}) \in \Phi \Leftrightarrow \exists i \in \{1, .., n\}, (N, \tilde{N}) \in \phi_i$.

If a *kit* is induced by a set of Petri net reduction rules, it is called a reduction kit. Likewise, if a *kit* is induced by a set of Petri net synthesis rules, it is called a synthesis kit.

We say that Φ (strongly) preserves ψ if and only if $\forall i \in \{1, .., n\}$, ϕ_i (strongly) preserves ψ.

Finally, Φ^* denotes the transitive closure of Φ, where $(N_0, N_m) \in \Phi^*$ if and only if there exists a sequence $(\phi_1, N_1), .., (\phi_m, N_m)$ such that $\forall\ i \in \{1, .., m\},\ (N_{i-1}, N_i) \in \phi_i$.

Lemma 4: Kit Property Preservation

Let ψ be a Petri net property, Φ a *kit*, which strongly preserves ψ, and N, $\tilde{N}$ be two Petri nets such that $(N, \tilde{N}) \in \Phi^*$, then one has:

- $N \models \psi \Leftrightarrow \tilde{N} \models \psi$

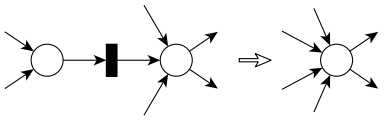
Suppose that Φ is a reduction *kit*, Φ is said to be a complete reduction *kit* with respect to the property ψ if and only if there exists an *atomic* Petri net N_{Atomic} (i.e. an irreducible Petri net) such that $N_{Atomic} \models \psi$ and $\forall\ N \models \psi,\ (N, N_{Atomic}) \in \Phi^*$. Conversely, suppose that Φ is a synthesis *kit*, Φ is said to be a complete synthesis *kit* with respect to the property ψ if and only if there exists an *atomic* Petri net N_{Atomic} such that $N_{Atomic} \models \psi$ and $\forall\ N \models \psi,\ (N_{Atomic}, N) \in \Phi^*$.

2.2.3.2/ Well-known reduction rules

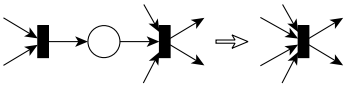
This section presents an overview of well-known reductions rules preserving boundedness and liveness, two properties closely related to the soundness of workflow nets (Section 2.2.2.4 page 34). Note that we restrict this section to rules whose *conditions of applications* are structurally defined and that are applicable to workflow nets.

In [Murata, 1989], six reduction rules strongly preserving boundedness and liveness are given. From those six rules only five are applicable to workflow nets. Figure 2.15 depicts these six reduction rules: Every net fragment shown on the left can be reduced to the fragment on the right while preserving boundedness and liveness.

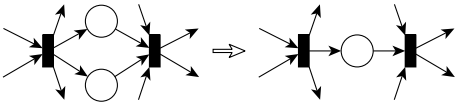
Note that many tools implement these reduction rules as a pre-processing step to analysis. For example, before the analysis of soundness, Woflan [Verbeek et al., 2000] – a Petri net based work-



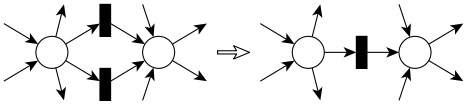
(a) Fusion of Series Places



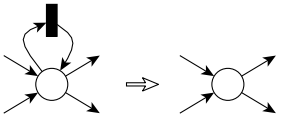
(b) Fusion of Series Transitions



(c) Fusion of Parallel Places



(d) Fusion of Parallel Transitions



(e) Elimination of Self-loop Places

Figure 2.15: Reduction rules of [Murata, 1989]

flow diagnosis tool – can apply these five reduction rules in order to reduce the size of the analysed workflow net.

In [Desel et al., 2005], three reduction rules preserving boundedness and liveness are presented and proved to be complete over the subclass of free-choice Petri nets. The first rule, denoted ϕ_A is depicted by Figure 2.16.

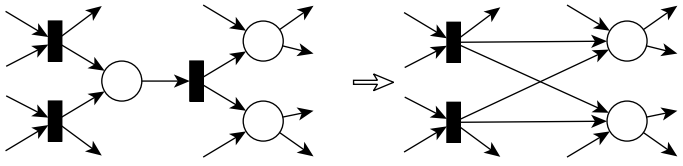


Figure 2.16: Reduction Rule ϕ_A of [Desel et al., 2005]

The second rule, denoted ϕ_S, states that a place p can be removed from a Petri net $N = \langle \mathcal{W} \rangle$ with at least two places if p is not an isolated place (i.e. ${}^\bullet p \cup p^\bullet \neq \emptyset$), and p is linearly dependent of a distinct set of places (i.e. there exists $\Lambda \in \mathbb{N}^{(1,|P|)}$ such that $\Lambda_{(1,p)} = 0$ and $\Lambda * \mathcal{W} = \mathcal{W}_{(p,_)}$ where $\mathcal{W}_{(p,_)}$ is the row of $\mathcal{W}$ corresponding to place p).

The third rule, denoted ϕ_T, states that a transition t can be removed from a Petri net $N = \langle \mathcal{W} \rangle$ with at least two transitions if t is not an isolated transition (i.e. ${}^\bullet t \cup t^\bullet \neq \emptyset$), and t is linearly dependent of a distinct set of transitions (i.e. there exists $\Lambda \in \mathbb{N}^{(|T|,1)}$ such that $\Lambda_{(t,1)} = 0$ and $\mathcal{W} * \Lambda = \mathcal{W}_{(_,t)}$ where $\mathcal{W}_{(_,t)}$ is the column of $\mathcal{W}$ corresponding to transition t).

As previously mentioned, this kit is a complete kit over the subclass of free-choice Petri Net with respect to boundedness and liveness. Thus, any bounded and live free-choice Petri nets can be reduced to an atomic Petri net defined as $N_{Atomic} = \langle\{p\}, \{t\}, \{(p,t),(t,p)\}\rangle$. As a result, we can use these rules on the k-closure of a free-choice workflow net to determine whether this free-choice workflow net is k-sound.

Numerous other authors have investigated reduction rules for Petri nets [Berthelot, 1987, Lee-Kwang et al., 1987], notably in the context of workflow nets [Voorhoeve et al., 1997, Sadiq et al., 2000, Lin et al., 2002, Lohmann et al., 2009, Hichami et al., 2014]. Furthermore, similar approaches have been applied to other languages such as EPCs [Kindler, 2004, van Dongen et al., 2005, van Dongen et al., 2007] and BPMN [Goldman et al., 2012]. Finally let us note that reduction rules have also been applied for the analysis of Petri net extensions such as time Petri nets [Sloan et al., 1996] and coloured workflow nets [Esparza et al., 2016].

2.3/ Constraint Systems

This section defines the notion of constraint system and provides an overview of the resolution techniques used to decide their satisfiability.

2.3.1/ Definition

Constraint systems model mathematical problems defined by a finite set of constraints (properties) over a finite set of variables with values ranging over their respective domains.

Formally, a constraint system is a tuple $\Omega =< X, D, C >$ where X is a set of variables $\{x_1, \ldots, x_n\}$, D is a set of domains $\{d_1, \ldots, d_n\}$, where d_i is the domain associated with the variable x_i, and C is a set of constraints $\{c_1(X_1), \ldots, c_m(X_m)\}$, where a constraint c_j involves a subset X_j of the variables of X.

An interpretation (also called a model) of a constraint system Ω is an assignment to every variable of a value from its domain. Formally, an interpretation is defined by a valuation function v assigning to each variable x_i of X a value from its domain d_i. An interpretation v satisfies (i.e. is consistent with respect to) a constraint $c(X)$ of C if the projection of v on X is in $c(X)$.

An interpretation v of a constraint system Ω is said to satisfy the constraint system Ω (i.e. v is a solution of Ω), noted $\Omega \models v$, if and only if v is an interpretation such that all the constraints C are satisfied. A constraint system Ω is said consistent or satisfiable if and only if there exists an interpretation v satisfying (i.e. consistent with respect to) all the constraints C.

Constraint systems thus model NP-complete problems as search problems where the corresponding search space is the Cartesian product space $d_1 \times \ldots \times d_n$.

Determining the consistency/satisfiability of constraint systems has been the focus of a lot of research work [Bordeaux et al., 2006, Biere et al., 2009, Oliveras, 2014].

In this thesis we restrict ourselves to the use of constraint systems over finite domains for which recent advances in resolution methods provide an efficient resolution regarding satisfiability.

Two approaches can be distinguished to solve this problem, namely Logic Programming and more specifically Constraint Logic Programming (CLP) over Finite Domains [Tsang, 1993] and Satisfiability Modulo Theories (SMT) over the theory of non-linear integer arithmetic [De Moura et al., 2011].

2.3.2/ Constraint Logic Programming (CSP)

Constraint satisfaction problems [Tsang, 1993] over finite domains are typically solved using a form of search. The most used techniques are based on variants of backtracking, constraint propagation, and local search.

A constraint system can be solved using generate-and-test paradigm (GT) that systematically generates each possible value assignment and then it tests whether it satisfies all the constraints. An other search paradigm, considered more efficient, is the backtracking paradigm (BT) that is the most commonly used algorithm. The backtracking paradigm is based on a recursive algorithm maintaining a partial interpretation. It proceeds as follows, first, all variables are unassigned. At each step, an unassigned variable is chosen, and all possible values are assigned to it in turn. For each value, the consistency of the partial interpretation with respect to all constraints is checked. If the partial interpretation is found consistent a recursive call is performed otherwise the algorithm *backtracks* to the previous variable. Whenever a complete interpretation is found consistent the algorithm can conclude that the constraint system is satisfiable.

The problem of most search algorithms based on backtracking is the occurrence of many *backtracks* to alternative choices, which degrades the efficiency of the system. Moreover, the efficiency of backtracking algorithms depends considerably on the order in which variables and their values are considered for instantiations. To cope with this issue various heuristics exist for dynamic or static ordering of values and variables. Further, several variants of backtracking exist. For example, *backjumping* [Dechter et al., 2002] allows saving part of the search by backtracking more than one variable in some cases. *Constraint learning* [Bayardo et al., 1996] infers and saves new constraints that can be later used to avoid part of the search. *Look-ahead* [Frost et al., 1995] attempts to foresee the effects of choosing a variable or a value, sometimes determining in advance whether a partial interpretation is consistent.

The late detection of inconsistency is the main disadvantage of GT and BT paradigms. Therefore constraint propagation techniques has been introduced to prune the search space. They are methods that enforce a form of local consistency, which are conditions related to the consistency of a group of variables and/or constraints turning a problem into one that is equivalent but is usually simpler to solve. The most known and used forms of local consistency are arc consistency, hyper-arc consistency, and path consistency [Kumar, 1992].

Well-known CSP solvers include Opturion CPX, OR-Tools, and SICStus Prolog [Carlsson et al., 1988].

Let us note that a lot of work has been done [Melzer et al., 1996, Desel, 1998, Schmidt, 2001, Soliman, 2008, Bourdeaud'Huy et al., 2008, Kleine et al., 2010, Wimmel et al., 2011] in order to model and to analyse the behaviour of Petri nets by using equational approaches that can be handled by constraint logic programming.

2.3.3/ Satisfiability Modulo Theories (SMT)

Constraint systems with variables over finite domains can be translated into first-order formulae over the theory of non-linear integer arithmetic so that their satisfiability can efficiently be checked using the satisfiability modulo theories approach [De Moura et al., 2011].

Satisfiability modulo theories consists of deciding the satisfiability of a first-order formula with respect to a background theory. Two main approaches are distinguished, the *eager approach* and the *lazy approach*.

The *eager approach* is based upon the translation of SMT instances into equisatisfiable Boolean SAT instances, so that off-the-shelf SAT solver can be used. The *lazy approach*, used by most SMT solvers, is based on a DPLL search with theory-specific solvers[Nieuwenhuis et al., 2006], called T-solvers, that handle conjunctions of predicates from a given theory T. In this approach boolean reasoning is handled by a DPLL-based SAT solver which interacts with a T-solver. The T-solver handles the theory T reasoning and check the feasibility of conjunctions of theory predicates passed on to it from the DPLL-based SAT solver as it explores the boolean search space of the formula.

Well-known SMT solvers include Z3 [De Moura et al., 2008] and CVC4 [Barrett et al., 2011].

In the context of Petri net analysis a lot of approaches using SMT resolution have been investigated [Monakova et al., 2009, Esparza et al., 2014, Pólrola et al., 2014, Esparza et al., 2015].

In this first part, we provided a review of some mathematical notions and background notations used throughout this thesis. We also reviewed some elements of the state of the art regarding Petri nets, workflow nets and existing approaches for their analysis as well as constraint systems together with approaches for their resolution. Based on these elements, the following part introduces the contributions made to this thesis.

II

Contributions

3

Verification of Modal specification

> "Being abstract is something profoundly different from being vague... The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise."
>
> — Edsger Dijkstra

Contents

3.1 Over Ordinary Workflow Nets 44
3.1.1 Extended Modal Specification 44
3.1.2 Modelling Executions of Workflow Nets 46
3.1.3 Verifying Extended Modal Specifications 53
3.2 Over Abstract Workflow Nets 55
3.2.1 Abstract Petri Nets 55
3.2.2 Extended Abstract Modal Specification 59
3.2.3 Modelling Executions of Abstract Workflow Nets 61
3.2.4 Verifying Extended Abstract Modal Specifications 64
3.3 Synthesis 65

To assist engineers in their specification and validation activities, modal specifications [Larsen, 1989] have been designed to allow *loose* specifications of models under development (Section 2.1.10 page 28). This chapter presents an approach to the verification of modal specifications.

The first section deals with ordinary workflow nets. It defines extended modal specifications, an extension of modal specifications that enables the description of complex modal properties expressing requirements on several transitions and on their causalities. It then describes an original modelling of workflow nets as constraint systems. The proposed modelling follows the well-known schema of *divide and conquer*. The main idea is to decompose modelled executions in segments defined over conflicts-free (i.e. marked graph) subnets, allowing the structural verification of their correctness in order to guaranty that the modelled executions are correct. It concludes by applying this modelling methodology to the verification of extended modal specification of workflow nets.

The second section deals with the generalisation of the approach defined in the first section to handle abstract Petri net, a suited abstract representation notably able to model ordinary, generalised and coloured Petri nets. It begins with the formal definition of abstract workflow nets, defining the largest model representation on which the modelling of executions presented in the first section can be extended. It then generalises the concept of extended modal specifications over ordinary workflow nets to the realm of abstract workflow nets. Based on these definitions it follows by presenting the constraint system used to model executions of abstract workflow nets as the composition of

segments verifiable structurally. Finally it exposes how abstract extended modal specifications can be verified on the basis of the previously described executions modelling.

The constraint based modelling approach we define presents several advantages compared to explicit or symbolic state space based verification techniques described in Section 2.2.1 page 29:

- The search is quite focussed from the beginning as we traverse the solution space of the state equation rather than the underlying semantic labelled transition systems of workflow nets;
- There is in most case no need to explicitly compute the ordering of all transitions composing executions, instead only the ordering of segments composing executions is required;
- The method may perform well when verifying the conformity of workflow nets with respect to *necessary* and *inadmissible* behaviours thanks to adequate use of over-approximations;
- The computation workload is shifted to existing mature and efficient constraint solving tools (Section 2.3 page 38).

The third and final section concludes this chapter by summarising the contributions made to the verification of modal specifications.

3.1/ Over Ordinary Workflow Nets

This section aims to present a verification method that enables the verification of modal specifications over ordinary workflow nets. It first defines extended modal specifications over ordinary workflow nets, an extension of modal specifications required to express requirements on several transitions and on their causalities. It then aims at defining a generic framework based on the modelling of ordinary workflow nets executions through constraint systems (Section 2.3 page 38). Finally, it presents how this generic framework is used to verify the (in)validity of extended modal specifications over ordinary workflow nets.

3.1.1/ Extended Modal Specification

Modal specifications of workflow nets (as presented in Section 2.1.10 page 28) allow specifiers to indicate that a specific transition is *necessary* or just *admissible*.

While basic modal specifications are useful, they usually lack expressiveness for real-life applications, as only individual transitions are concerned with.

Example 9: Illustration of the limits of basic modal specifications

Consider the on-demand order delivery workflow net described in Section 2.1.6 page 21. One would like to specify that a modal property of this workflow net is the fact that a client must pay his order either using a credit card or by cash. However, as this requirement concerns behaviour involving two distinct transitions, it cannot be expressed using basic modal specifications as presented in Section 2.1.10.

Therefore, we propose to extend modal specifications to express requirements on several transitions and on their causalities.

To this end, the language $S(N)$ of well-formed modal specification formulae associated to a workflow net $N = \langle P, T, F \rangle$ is defined as follows.

Definition 33: Well-formed Modal Specification Formulae

Let $N = \langle P, T, F \rangle$ be a workflow net where $T = \{t_1, .., t_n\}$. The language $S(N)$ of well-formed modal specification formulae associated to a workflow net N is defined by the following grammar of axiom A:

$$A \rightarrow (A \wedge A) \mid (A \vee A) \mid (\neg A) \mid B$$
$$B \rightarrow t_1 \mid .. \mid t_n$$

The domain of a well-formed modal specification formula m, denoted $Domain(m)$, is the set of transitions appearing in the formula (i.e. the leafs of the abstract syntax tree of m).

These well-formed modal specification formulae allow specifiers to express modal properties about workflow nets correct executions.

Let $\sigma \in \Sigma_1$ be a correct execution of N and m a well-formed modal specification formula, we denote $\sigma \models m$ the fact that the modal property expressed by m is satisfied by the execution σ. Formally, given $t \in T$ and $A_1, A_2 \in S(N)$, we have $\sigma \models t \Leftrightarrow O_t(\sigma) > 0$, $\sigma \models (A_1 \wedge A_2) \Leftrightarrow \sigma \models A_1 \wedge \sigma \models A_2$, $\sigma \models (A_1 \vee A_2) \Leftrightarrow \sigma \models A_1 \vee \sigma \models A_2$, and $\sigma \models (\neg A_1) \Leftrightarrow \neg(\sigma \models A_1)$.

Similarly to the way a transition can be characterized as a *may*-transition or a *must*-transition, any modal specification formula $m \in S(N)$ can be interpreted as a *may*-formula or a *must*-formula.

On the one hand, a *may*-formula describes a modal property that has to be ensured by at least one correct execution of the considered workflow net. On the other hand, a *must*-formula describes a modal property that has to be ensured by all the correct executions of the considered workflow net.

Further, given a well-formed *may*-formula (resp. *must*-formula) $m \in S(N)$, the workflow N satisfies m, written $N \models_{may} m$ (resp. $N \models_{must} m$), when at least one (resp. all) correct execution(s) of N satisfies (resp. satisfy) the modal property expressed by m.

Formally, given $m \in S(N)$:

$$N \models_{may} m \Leftrightarrow \exists\, \sigma \in \Sigma_1, \sigma \models m, \text{ and}$$
$$N \models_{must} m \Leftrightarrow \forall\, \sigma \in \Sigma_1, \sigma \models m.$$

Note that the set of *may*-formulae forms a subset of Computation Tree Logic (CTL) formulae interpreted over finite traces where only the *possibly* operator (i.e. along at least one path) is used. Likewise the set of *must*-formulae forms a subset of CTL formulae where only the *inevitably* operator (i.e. along all paths) is used [Clarke et al., 1986, De Giacomo et al., 2013, Westergaard, 2011].

Example 10: Illustration of an extended modal specification formula

To illustrate the use of extended modal specification formula let N be the on-demand order delivery workflow net considered in Example 9. The fact that a client must pay his order either using a credit card or by cash can be specified by the following *must*-formula: $m = Client\ Pays\ By\ Cash \vee Client\ Pays\ By\ Credit\ Card$ whose domain is $Domain(m) = \{Client\ Pays\ By\ Cash, Client\ Pays\ By\ Credit\ Card\}$. It follows that, in order to be valid with respect to this specification, N has to satisfy the *must*-formula m (i.e. $N \models_{must} m$). Note that, N does indeed satisfy the *must*-formula m.

The previous definitions of the syntax and semantic of *may*-formulae and *must*-formulae allow us to define what we call an extended modal specification as follows.

Definition 34: Extended Modal Specifications

Let $N = \langle P, T, F \rangle$ be a workflow net. An extended modal specification of N is a tuple $\langle a_{must}, A_{may} \rangle$ where:

- a_{must} is a *must*-formula, and
- A_{may} is a set of *may*-formulae.

Note that, in this definition, only a single *must*-formula is needed while a set of *may*-formulae is needed. This is due to the fact that given $A_1, A_2 \in S(N)$, $N \models_{must} A_1 \wedge N \models_{must} A_2 \Leftrightarrow N \models_{must} (A_1 \wedge A_2)$ whereas $N \models_{may} A_1 \wedge N \models_{may} A_2 \nRightarrow N \models_{may} (A_1 \wedge A_2)$.

Given a workflow net N, we say that N satisfies (i.e. is conform to) an extended modal specification $\langle a_{must}, A_{may} \rangle$, noted $N \models \langle a_{must}, A_{may} \rangle$, if and only if:

$$N \models_{must} a_{must}, \text{ and}$$

$$\forall\, a_{may} \in A_{may}, N \models_{may} a_{may}.$$

We defined extended modal specifications over workflow nets, an extension of modal specifications that enables the description of complex modal properties expressing *necessary* or *admissible* behaviour involving several transitions and their causalities. The next section describes constraint systems which aim at modelling correct workflow net executions in order to enable the verification of such extended modal specifications.

3.1.2/ Modelling Executions of Workflow Nets

This section exposes the modelling of correct executions of ordinary workflow nets through their decomposition into segments verifiable structurally.

We begin by considering the well-known *state equation*, also known as the *fundamental equation* (Section 2.2.2.1 page 30). The *state equation* has been used earlier for verification purposes. It has been notably considered in the analysis of safety properties [Esparza et al., 2000] as well as reachability [Schmidt, 2001, Wimmel et al., 2011].

Definition 35: State Equation Constraint System

Let $N = \langle P, T, F \rangle$ be a workflow net and M_a, M_b two markings of N, the state equation constraint system $\mathcal{S}(N, M_a, M_b)$, associated with it, is:

- $\forall p \in P,\ \nu(p) = \sum_{t \in p^\bullet} \nu(t) + M_b(p) = \sum_{t \in {}^\bullet p} \nu(t) + M_a(p)$

where $\nu : P \cup T \to \mathbb{N}$ is a valuation function.

The solutions of the constraint system of Definition 35 are related to the execution of N as stated by Theorem 10.

Theorem 10: Reachability Necessary Condition

Let $N = \langle P, T, F \rangle$ be a workflow net. If $M_a \xrightarrow{*} M_b$ then there exists a valuation satisfying $\mathcal{S}(N, M_a, M_b)$.

Proof. Suppose $M_a \xrightarrow{*} M_b$, by definition there exists $\sigma = t_1, t_2, ..., t_n$ such that $M_a \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 \xrightarrow{t_3} .. \xrightarrow{t_{n-1}} M_{n-1} \xrightarrow{t_n} M_b$.

Let $\nu : P \cup T \to \mathbb{N}$ be defined as follows:

- $\forall t \in T,\ \nu(t) = O_t(\sigma)$
- $\forall p \in P,\ \nu(p) = \sum_{j \in \{1,2,...,n-1\} \cup \{a,b\}} M_j(p)$

As the workflow N is an ordinary Petri net, the sum of the tokens in all markings of a place is equal to the sum of the occurrences of transitions producing (resp. consuming) a token at this place plus the number of token(s) in marking M_b (resp. M_a).

Therefore $\forall p \in P,\ \sum_{j \in \{1,2,...,n-1\} \cup \{a,b\}} M_j(p) = \sum_{t \in p^\bullet} O_t(\sigma) + M_b(p) = \sum_{t \in {}^\bullet p} O_t(\sigma) + M_a(p)$.

By definition of ν, it follows that $\forall p \in P,\ \nu(p) = \sum_{t \in p^\bullet} \nu(t) + M_b(p) = \sum_{t \in {}^\bullet p} \nu(t) + M_a(p)$.

Consequently, ν is a valuation satisfying $\mathcal{S}(N, M_a, M_b)$. □

More precisely, the constraint system $\mathcal{S}(N, M_a, M_b)$ models the fact that for each place p of N, the number of token(s) entering p plus the number of token(s) in $M_a(p)$ is equal to the number of tokens leaving p plus the number of token(s) in $M_b(p)$. Let ν be a valuation function satisfying $\mathcal{S}(N, M_a, M_b)$, ν aims at modelling a transition sequence σ such that $\forall t \in T, O_t(\sigma) = \nu(t)$. However, there is no guarantee that there exists such σ corresponding to an execution of N reaching the marking M_b from the marking M_a. Solutions of $\mathcal{S}(N, M_a, M_b)$ which do not correspond to an execution of N are called spurious solutions. Indeed, spurious solutions can appear because the order of transition firing is not taken into account in the modelled execution.

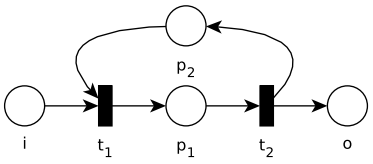


Figure 3.1: Workflow net (N_1) used to illustrate a spurious solution of the *state equation*

Example 11: Illustration of a spurious solution of $\mathcal{S}$

To illustrate a spurious solution of $\mathcal{S}$ let us consider the workflow N_1 depicted by Figure 3.1. In this workflow, the transition t_2 has to consume a token produced by t_1 in p_1, likewise transition t_1 has to consume a token produced by t_2 in p_2. As two transitions cannot *fire* simultaneously from an initial marking M_i, transitions t_1 and t_2 are *dead* transitions. It follows that there exists no correct execution of N_1 (i.e. $\nexists\, \sigma, M_i \xrightarrow{\sigma} M_o$). However, the following valuation ν_1 such that $\forall\ n \in \{i, p_1, p_2, o, t_1, t_2\}, \nu_1(n) = 1$ is a valuation satisfying $\mathcal{S}(N_1, M_i, M_o)$ that does not correspond to any correct execution of N_1 and is therefore a spurious solution.

Nevertheless, as stated by Theorem 10, if there is no valuation satisfying $\mathcal{S}(N, M_a, M_b)$ then the marking M_b is not reachable from marking M_a. In this sense the constraint system $\mathcal{S}(N, M_a, M_b)$ defines an over-approximation of all the executions of N reaching the marking M_b from the marking M_b.

In order to dismiss the spurious solutions of $\mathcal{S}$, we propose to refine it. To do so, we consider the modelled executions subnets structural features.

Let ν be a solution of the constraint system $\mathcal{S}(N, M_a, M_b)$, the subnet associated with it is an ordinary Petri net such that only the places and transitions of N involved by the valuation ν are considered. Note that we also remove the places which have a greater or equal number of tokens in the marking M_a (resp. M_b) with respect to the number of tokens consumed from (resp. produced in) those places.

Definition 36: State Equation Solution Subnet

Let $N = \langle P, T, F\rangle$ be a workflow net, M_a, M_b two markings of N, $\nu : P \cup T \to \mathbb{N}$ a satisfying valuation of $\mathcal{S}(N, M_a, M_b)$, $\mathcal{U}^+ : P \mapsto \mathbb{N}$ a function such that $\forall p \in P$, $\mathcal{U}^+(p) = \sum_{t \in p^\bullet} \nu(t)$ and $\mathcal{U}^- : P \mapsto \mathbb{N}$ a function such that $\forall p \in P$, $\mathcal{U}^-(p) = \sum_{t \in {}^\bullet p} \nu(t)$. We define the subnet $sN(\nu) = \langle sP, sT, sF\rangle$ as an ordinary Petri net where:

- $sP = \{p \in P \mid \nu(p) > 0 \wedge (\mathcal{U}^+(p) > M_a(p) \vee \mathcal{U}^-(p) > M_b(p))\}$
- $sT = \{t \in T \mid \nu(t) > 0\}$, and
- $sF = \{(a, b) \in F \mid a \in (sP \cup sT) \wedge b \in (sP \cup sT)\}$

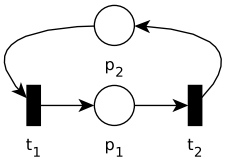


Figure 3.2: Illustration of a state equation solution subnet ($sN(\nu_1)$)

Example 12: Illustration of a subnet associated with a valuation of $\mathcal{S}$

To illustrate this construction, Figure 3.2 depicts the subnet $sN(\nu_1)$ associated to the valuation ν_1 satisfying $\mathcal{S}(N_1, M_i, M_o)$ defined in Example 11.

By Lemma 1 page 33, we can infer that the subnet $sN(\nu)$ of a non spurious solution ν of $\mathcal{S}(N, M_a, M_b)$ does not contain a trap nor a siphon. Indeed, as a marked trap cannot be unmarked and places marked in the final marking M_b have been removed, $sN(\nu)$ cannot contain a trap. Likewise as an unmarked siphon cannot be marked and places of $sN(\nu)$ must have at least a token consumed, $sN(\nu)$ cannot contain a siphon.

Example 13: Illustration of the refutation of a spurious valuation of $\mathcal{S}$

This allows us to directly conclude that ν_1, as defined in Example 11, is a spurious solutions of $\mathcal{S}(N_1, M_i, M_o)$ since $sN(\nu_1)$ contains a trap $\{p_1, p_2\}$.

An interesting property of the subnet $sN(\nu)$ of any solution ν of $\mathcal{S}(N, M_a, M_b)$ relates the presence of siphons to the presence of traps, and vice-versa, as stated by Theorem 11.

Theorem 11: Siphon/Trap Property of Subnet

Let $N = \langle P, T, F\rangle$ be a workflow net, M_a, M_b two markings of N, $\nu : P \times T \to \mathbb{N}$ a valuation satisfying $\mathcal{S}(N, M_a, M_b)$ and $sN(\nu)$ the subnet associated with ν.
If $sPN(\nu)$ contains a trap (resp. siphon) G then G is also a siphon (resp. trap).

Proof. ($\Rightarrow$) Let $G \subseteq sP$ such that $G \neq \emptyset$, by definition of $\mathcal{S}(N, M_a, M_b)$ and ν we have $\sum_{p\in G} \nu(p) = \sum_{p\in G} \sum_{t\in p^\bullet} \nu(t) = \sum_{p\in G} \sum_{t\in {}^\bullet p} \nu(t)$. It implies $\sum_{p\in G} \sum_{t\in p^\bullet \cap G^\bullet} \nu(t) + \sum_{p\in N} \sum_{t\in p^\bullet \cap sT/G^\bullet} \nu(t) = \sum_{p\in N} \sum_{t\in {}^\bullet p\cap G^\bullet} \nu(t) + \sum_{p\in N} \sum_{t\in {}^\bullet p\cap sT/G^\bullet} \nu(t)$ that can be simplified as $\sum_{p\in N} \sum_{t\in p^\bullet} \nu(t) = \sum_{p\in N} \sum_{t\in {}^\bullet p\cap G^\bullet} \nu(t) + \sum_{p\in N} \sum_{t\in {}^\bullet p\cap sT/G^\bullet} \nu(t)$ because $\forall p \in N. p^\bullet \cap sT/G^\bullet = \emptyset$. Let G be a trap (i.e. $G^\bullet \subseteq {}^\bullet G$) such that G is not a siphon (i.e. ${}^\bullet G \not\subseteq G^\bullet$). Thus, one has $\sum_{p\in N} \sum_{t\in p^\bullet} \nu(t) = \sum_{p\in N} \sum_{t\in {}^\bullet p\cap G^\bullet} \nu(t)$ implying $\sum_{p\in N} \sum_{t\in p^\bullet} \nu(t) = \sum_{p\in N} \sum_{t\in p^\bullet} \nu(t) + \sum_{p\in N} \sum_{t\in {}^\bullet p\cap sT/G^\bullet} \nu(t)$. We finally have $\forall p \in N. {}^\bullet p \cap sT/G^\bullet = \emptyset$ because $\forall t \in sT. \nu(t) > 0$. This implies ${}^\bullet G \subseteq G^\bullet$, a contradiction.
($\Leftarrow$) The proof that if G is a siphon then G is a trap, is similar. □

Example 14: Illustration of the property stated by Theorem 11

This property is verified by $sN(\nu_1)$, the valuation defined in Example 11, as the trap $\{p_1, p_2\}$ identified in Example 13 is also a siphon.

According to Theorem 11, one can decide the presence of siphons and traps in a subnet by deciding about the presence of siphons only (or equivalently to the presence of traps only). Alternatively, one can decide the presence of siphons and traps in a subnet by deciding about the presence of P-semiflow (Lemma 3 page 33).

The presence of siphons in an ordinary Petri net can be decided by evaluating the satisfiability of the following constraint system.

Theorem 12: Siphon Presence Constraint System

Let $N = \langle P, T, F\rangle$ be an ordinary Petri net and $\mathcal{B}(N)$ the constraint system defined as follows:

- $\forall p \in P, \forall t \in {}^\bullet p. \sum_{p'\in {}^\bullet t} \xi(p') \geq \xi(p)$
- $\sum_{p\in P} \xi(p) > 0$

where $\xi : P \to \{0, 1\}$ is a valuation function.
N contains a siphon if and only if there is a valuation satisfying $\mathcal{B}(N)$.

Proof. ($\Leftarrow$) Let ξ be a valuation satisfying $\mathcal{B}$, and $G \subseteq P$ such that $\xi(p) = 1 \Leftrightarrow p \in G$. Then $\forall p \in G, \forall t \in {}^\bullet p, \sum_{p'\in {}^\bullet t} \xi(p') \geq 1$, which implies ${}^\bullet G \subseteq G^\bullet$. Consequently, G is a siphon.

($\Rightarrow$) Suppose that G is a siphon (i.e. ${}^\bullet G \subseteq G^\bullet$) then the valuation $\xi(p)$ defined as $\xi(p) = 1$ if $p \in G$, and 0 otherwise, satisfies $\mathcal{B}(N)$. □

Using the constraint system $\mathcal{S}$ together with the constraint system $\mathcal{B}$ leads to the definition of the constraint system $\mathcal{Q}$ as follows.

Definition 37: State Equation + Absence of Siphon Constraint System

Let $N = \langle P, T, F\rangle$ be a workflow net and M_a, M_b two markings of N, the constraint system $\mathcal{Q}(N, M_a, M_b)$, associated with it, is:

- $\mathcal{S}(N, M_a, M_b) \models \nu$
- $\nexists\, \xi$ such that $\mathcal{B}(sN(\nu)) \models \xi$

where $\nu : P \cup T \to \mathbb{N}$ is a valuation function.

While many of the spurious solutions of $\mathcal{S}(N, M_a, M_b)$ are discarded from the solutions of $\mathcal{Q}(N, M_a, M_b)$, the latter constraint system still defines an over-approximation of the valid execution of N leading to M_b from M_a. This is due to the fact that the absence of traps and siphons in the subnets of solutions of $\mathcal{Q}(N, M_a, M_b)$ is only a necessary but not a sufficient condition to the existence of a valid execution of N leading to M_b from M_a.

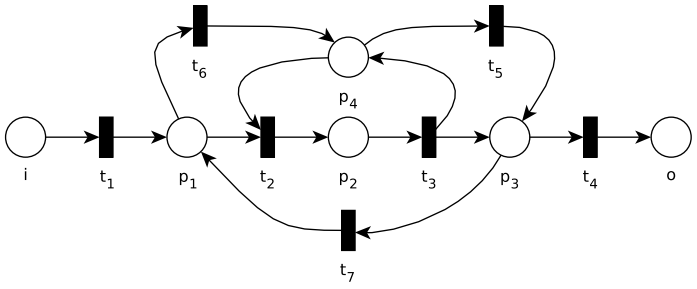


Figure 3.3: Workflow net (N_2) used to illustrate a spurious solution of $\mathcal{Q}$

Example 15: Illustration of a spurious solution of $\mathcal{Q}$

Consider the workflow N_2 depicted in Figure 3.3.
The valuation ν_2, where $\forall\ n \in \{i, p_2, o, t_1, t_2, t_3, t_4, t_5, t_6, t_7\}, \nu_2(n) = 1$ and $\forall\ n \in \{p_1, p_3, p_4\}, \nu_2(n) = 2$, is a valuation satisfying $\mathcal{Q}(N_2, M_i, M_o)$. However, the transition t_2 of this workflow N_2 is a dead transition from the initial marking M_i. Indeed, to be enabled, the transition t_2 requires the presence of two tokens (one in place p_1 and one in place p_4), but the initial marking only has one token and the only transition creating a token is t_3 which requires a token to be produced by t_2 in place p_2. Therefore there exists no execution σ such that $M_i \xrightarrow{\sigma} M_o$ and $O_{t_2}(\sigma) > 0$. It follows that ν_2 is a spurious solutions of $\mathcal{Q}(N_2, M_i, M_o)$.

While defining an over-approximation might be useful for the verification of safety properties, in our case, we want to be able to models any executions of a workflow net with enough precision in order to verify extended modal specifications.

To further refine $\mathcal{Q}$ so that sufficient conditions can be defined structurally, additional constraints are needed. We propose to consider solution ν of $\mathcal{Q}(N, M_a, M_b)$ such that $sN(\nu)$ is a marked graph (Section 2.1.4). The resulting constraint system is denoted $\mathcal{D}$.

Definition 38: State Equation + Absence of Siphon + MG Constraint System

Let $N = \langle P, T, F\rangle$ be a workflow net and M_a, M_b two markings of N, the constraint system $\mathcal{D}(N, M_a, M_b)$, associated with it, is:

- $\forall p \in P,\ \sum_{t \in {}^\bullet p} \nu(t) \leq 1 \wedge \sum_{t \in p^\bullet} \nu(t) \leq 1$
- $\mathcal{Q}(N, M_a, M_b) \models \nu$

where $\nu : P \cup T \to \mathbb{N}$ is a valuation function.

The solutions of the constraint system of Definition 38 are related to the execution of a workflow net N as stated by Theorem 13.

Theorem 13: Path Existence

Let $N = \langle P, T, F \rangle$ be a workflow net and M_a, M_b two markings of N. If there exists a valuation $\nu : P \cup T \rightarrow \mathbb{N}$ such that $\mathcal{D}(N, M_a, M_b) \models \nu$ then there exists a transition sequence σ such that $M_a \xrightarrow{\sigma} M_b$ and $\nu \models \sigma$.

Proof. Suppose there exists a valuation $\nu : P \cup T \rightarrow \mathbb{N}$ such that $\mathcal{D}(N, M_a, M_b) \models \nu$. By definition of $\mathcal{D}(N, M_a, M_b)$, $sN(\nu)$ is a marked graph with no siphons nor traps. It follows from Theorem 5 that $sN(\nu) = \langle sP, sT, sF \rangle$ is a live marked graph from any marking (in particular from an initial marking $M_\emptyset$ such that $\forall\ p \in sP, M_\emptyset(p) = 0$). Therefore, $\forall\ p \in sP$ there exists $t_{pre} \in {}^\bullet p$, $t_{post} \in p^\bullet$ such that $\nu(t_{pre}) > 0 \wedge \nu(t_{post}) > 0$ and $\forall\ \sigma$ such that $M_\emptyset \xrightarrow{\sigma} M_\emptyset$ is an execution of $sN(\nu)$ we have $O_{t_{post}}(\sigma) = 1 \Rightarrow O_{t_{pre}}(\sigma) = 1$. By induction, it follows that there exists σ such that $M_\emptyset \xrightarrow{\sigma} M_\emptyset$ an execution of $sN(\nu)$ where $\forall t \in T$ such that $\nu(t) > 0$, we have $O_t(\sigma) = 1$. We conclude that there exists σ such that $M_a \xrightarrow{\sigma} M_b$ is an execution of N where $\nu \models \sigma$. □

As stated by Theorem 13, any solution of $\mathcal{D}(N, M_a, M_b)$ models at least one valid execution of N leading to M_b from M_a. Note the solutions of $\mathcal{D}$ are abstractions of the executions they model as the ordering of the transitions they involve is not explicitly computed. However, as N might not belong to the subclass of marked graph, not all valid executions of N leading to M_b from M_a can be modelled by solutions of $\mathcal{D}(N, M_a, M_b)$. In this sense, the solutions of $\mathcal{D}(N, M_a, M_b)$ define an under-approximation of the valid executions of N leading to M_b from M_a.

We call segment every execution modelled by the constraint system $\mathcal{D}$. We now propose to decompose any execution of a workflow net into such segments. The resulting constraint system is denoted $\mathcal{U}$.

Definition 39: k-segment Execution Constraint System

Let $N = \langle P, T, F \rangle$ be a workflow net, M_a, M_b two markings of N, and $k \in \mathbb{N}$ a number of segments, the constraint system $\mathcal{U}(N, M_a, M_b, k)$, associated with it, is:

- $\exists\ M_1, \nu_1, \mathcal{D}(N, M_a, M_1) \models \nu_1$
- $\forall\ i \in \{2, .., k-1\}, \exists\ M_i, \nu_i, \mathcal{D}(N, M_{i-1}, M_i) \models \nu_i$
- $\exists\ \nu_k, \mathcal{D}(N, M_{k-1}, M_b) \models \nu_k$
- $\forall\ n \in P \cup T, \nu(n) = \sum_{i \in \{1,..,k\}} \nu_i(n)$

where $\nu : P \cup T \rightarrow \mathbb{N}$ is a valuation function.

The solutions of the constraint system of Definition 39 are related to the execution of a workflow net N as stated by Theorem 14.

Theorem 14: Path Existence

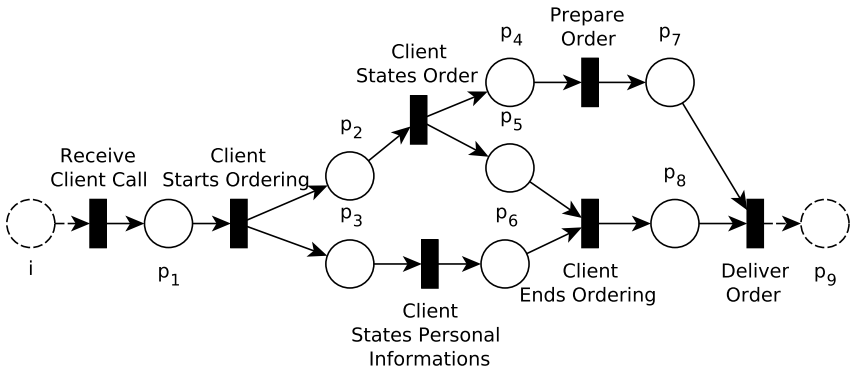
Let $N = \langle P, T, F \rangle$ be a workflow net and M_a, M_b two markings of N. There exists a transition sequence σ such that $M_a \xrightarrow{\sigma} M_b$ if and only if there exist $k \in \mathbb{N}$ a number of segments and $\nu : P \cup T \rightarrow \mathbb{N}$ a valuation function such that $\mathcal{U}(N, M_a, M_b, k) \models \nu$ and $\nu \models \sigma$.

Proof. ($\Rightarrow$) Suppose there exists a transition sequence $\sigma = t_1, .., t_k$ such that $M_a \xrightarrow{\sigma} M_b$.

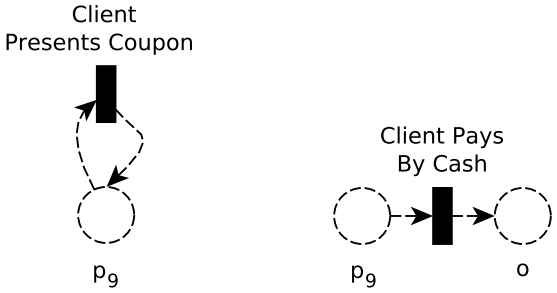
By definition there exists $M_1, .., M_{k-1}$ such that $M_a \xrightarrow{t_1} M_1..M_{k-1} \xrightarrow{t_k} M_b$. As the firing of a single transition t can be modelled by a valuation $\nu_t : P \cup T \mapsto \mathbb{N}$ satisfying $\mathcal{S}$ and the associated subnet $sN(\nu_t)$ is a marked graph without siphon composed of only transition t, we can infer that ν_t is also a valuation satisfying $\mathcal{D}$. It follows that there exists $\nu_1, .., \nu_k$ such that $\mathcal{D}(N, M_a, M_1) \models \nu_1, .., (N, M_{k-1}, M_b) \models \nu_k$. Let $\nu : P \cup T \mapsto \mathbb{N}$ be a valuation function such that $\forall\, n \in P, \nu(n) = \sum_{i \in \{1,..,k\}} \nu_i(n)$. By definition of $\mathcal{U}$, we can conclude that $\mathcal{U}(N, M_a, M_b, k) \models \nu$ and $\nu \models \sigma$.

($\Leftarrow$) Follows from Theorem 13. □

By Theorem 14, every execution composed of at most k segments and leading to a marking M_b from a marking M_a of a workflow net N can be modelled by the constraint system $\mathcal{U}(N, M_a, M_b, k)$.



(a) First segment (ν_{first})



(b) Second segment (ν_{second}) (c) Third segment (ν_{third})

Figure 3.4: Illustration of an execution modelled by three segments

Example 16: Illustration of an execution modelled by $\mathcal{U}$

Let us consider the on-demand order delivery workflow net $N = \langle P, T, F \rangle$ described in Section 2.1.6 page 21. Figures 3.4(a), 3.4(b), and 3.4(c) depict three segments respectively modelled by the valuations ν_{first}, ν_{second} and ν_{third}. The valuation ν where $\forall\, n \in P \cup T, \nu(n) = \nu_{first}(n) + \nu_{second}(n) + \nu_{third}(n)$ is a solution of the constraint system $\mathcal{U}(N, M_i, M_b, 3)$ and therefore a model of a correct execution of N. The places and transitions present in each figure correspond to the places and transition which valuation is equal to 1. The dashed elements are the elements not present in the subnet associated with the considered valuations. This illustrates how workflow nets complete executions can be broken down into pieces we call segments whose validity is guaranteed by the structural analysis of their subnets.

We described how ordinary workflow nets correct execution can by modelled through the definition of constraint systems. More precisely we showed how every execution could be modelled by segments verifiable structurally. The next section describes how the previously defined modelling can be used to verify extended modal specifications over ordinary workflow nets.

3.1.3/ Verifying Extended Modal Specifications

Let us recall that modal specifications enable the description of complex modal properties expressing *admissible* or *necessary* behaviour involving several transitions and their causalities. Such *admissible* and *necessary* behaviours are respectively defined by *may*-formulae and *must*-formulae.

Intuitively, a workflow net N is valid with respect to a *may*-formula m if and only if there exists a correct execution σ of N such that the modal property expressed by m is satisfied by the execution σ (i.e. $N \models_{may} m \Leftrightarrow \exists\, \sigma \in \Sigma_1, \sigma \models m$). Likewise a workflow net N is valid with respect to a *must*-formula m if and only if there does not exist a correct execution σ of N such that the modal property expressed by $\neg m$ is satisfied by the execution σ (i.e. $N \models_{must} m \Leftrightarrow \nexists\, \sigma \in \Sigma_1, \sigma \models (\neg m)$).

When determining whether or not a workflow net satisfies the modal properties of interest, we distinguish two decision problems. The first one, called the *K-bounded validity of a modal formula*, only considers executions formed by K segments, at most. The second one, called the *unbounded validity of a modal formula*, deals with executions formed by an arbitrary number of segments; it generalises the first problem.

In our approach, verifying modal formulae as defined in Section 3.1.1 page 44 relies on their expression by constraints.

Given $N = \langle P, T, F \rangle$ a workflow net and $m \in S(N)$ a modal property, we build the associated constraint system, noted $C(N, m)$, by replacing every terminal symbol $t \in T$ of m by $\nu(t) > 0$, where $\nu : P \cup T \mapsto \mathbb{N}$ is a valuation function.

Example 17: Illustration of the expression by constraints of a modal formula

Let N be the on-demand order delivery workflow net described in Section 2.1.6 page 21. The modal property m which states the fact that a client pays his order either using a credit card or by cash can be defined as $m = \textit{Client Pays By Cash} \vee \textit{Client Pays By Credit Card}$. By replacing every terminal symbol $t \in T$ of m by $\nu(t) > 0$, we obtain the following constraint system: $C(N, m) = \nu(\textit{Client Pays By Cash}) > 0 \vee \nu(\textit{Client Pays By Credit Card}) > 0$ where $\nu : P \cup T \mapsto \mathbb{N}$ is the valuation function.

Upon the expression of a modal property by a constraint system, we build a constraint system extending the constraint system $\mathcal{U}(N, M_i, M_o, k)$ which models the correct execution of N composed of at most k segments.

Definition 40: k-segment Modal Execution Constraint System

Let $N = \langle P, T, F \rangle$ be a workflow net, $k \in \mathbb{N}$ a number of segments, and $m \in S(N)$ a modal property, the associated constraint system $\mathcal{V}(N, k, m)$ is defined as follow:

- $\mathcal{U}(N, M_i, M_o, k) \models \nu$
- $C(N, m) \models \nu$

where $\nu : P \cup T \to \mathbb{N}$ is a valuation function.

The solutions of the constraint system of Definition 40 are related to the execution of a workflow net N as stated by Theorem 15.

Theorem 15: Path Existence

Let $N = \langle P, T, F \rangle$ be a workflow net and $m \in S(N)$ a modal property.
There exists a transition sequence σ such that $M_i \xrightarrow{\sigma} M_o$ and $\sigma \models m$ if and only if there exists $k \in \mathbb{N}$ and $\nu : P \cup T \to \mathbb{N}$ a valuation function such that $\mathcal{V}(N, k, m) \models \nu$, and $\nu \models \sigma$.

Proof. Follows from Theorem 14. □

By Theorem 15, there exists a correct execution of N composed of at most k segments satisfying a modal property $m \in S(N)$ if and only if there exists a valuation satisfying the constraint system $\mathcal{V}(N, k, m)$. It follows that we can determine the *K-bounded validity* of a modal property m, by determining the satisfiability of $\mathcal{V}(N, K, m)$. Consequently let m be a *may*-formula (resp. a *must*-formula), the workflow net N is said to be K-bounded valid with respect to m if and only if $\exists\, \nu, \mathcal{V}(N, K, m) \models \nu$ (resp. $\nexists\, \nu, \mathcal{V}(N, K, \neg m) \models \nu$).

Theorem 16: *K-bounded validity* of EMS

Let $N = \langle P, T, F \rangle$ be a workflow net and $\langle a_{must}, A_{may} \rangle$ be an extended modal specification of N. The correct executions of N composed of at most k segments satisfy the extended modal specification $\langle a_{must}, A_{may} \rangle$, noted $N \models_k \langle a_{must}, A_{may} \rangle$, if and only if:

- $\nexists\, \nu, \mathcal{V}(N, \neg a_{must}) \models \nu$, and
- $\forall\, a_{may} \in A_{may}, \exists\, \nu, \mathcal{V}(N, k, a_{may}) \models \nu$.

Proof. Follows from Theorem 15. □

By Theorem 16, the *K-bounded validity* of an extended modal specification can de determined through the evaluation of the satisfiability of the corresponding constraint system.

We continue by demonstrating that for K sufficiently large, the *K-bounded validity* of an extended modal specification is equivalent to its *unbounded validity*.

Theorem 17: Existence of K_{max}

Let $N = (P, T, F)$ be a workflow net, $\bar{R}_{must}$ the set of all well-formed *must*-formulae not satisfied by N, and R_{may} the set of all well-formed *may*-formulae satisfied by N. There exists K_{max} such that:

- $\forall f \in \bar{R}_{must}, \exists\, \nu, k \leq K_{max}, \mathcal{V}(N, k, \neg m) \models \nu$, and
- $\forall f \in R_{may}, \exists\, \nu, k \leq K_{max}, \mathcal{V}(N, k, m) \models \nu$.

Proof. Sketch. The set of correct executions of a workflow net is possibly infinite. This is due to the fact that T-invariants (i.e. sequence of transitions σ such that $M \xrightarrow{\sigma} M$) could be fired indefinitely. However, when considering the verification of modal formulae, we are only interested in the presence or absence of transitions in correct executions (i.e. the number of their firings does not matter). Therefore considering the set of correct executions where T-invariants are allowed to

fire at most once is enough to check the validity of modal formulae. This restricted set of correct executions is finite. As a consequence, there exists K_{max} such that every execution of this set can be modelled by K_{max} segments, at most. □

We presented an approach enabling the verification of (extended) modal specifications over ordinary workflow nets through the modelling of their executions by constraint systems. In the next section, this approach is generalised to the realm of abstract Petri nets, an abstract notion of Petri nets including ordinary, generalised and coloured Petri nets.

3.2/ Over Abstract Workflow Nets

The goal of this section is to generalise the verification method of modal specifications presented in the previous section to the realm of abstract workflow nets, a suited abstract representation notably able to model ordinary, generalised and coloured Petri nets. After formally defining abstract workflow nets as well as their abstract extended modal specification, this section generalises the modelling framework presented in the previous section to enable its use over abstract workflow nets. It then describes how this modelling framework is used to verify the (in)validity of extended modal specifications over abstract workflow nets.

3.2.1/ Abstract Petri Nets

This section introduces the notion of Abstract Petri nets (APN) which provides a generalisation of Petri nets (Section 2.1.2 page 11), and of coloured Petri nets (Section 2.1.5.2 page 19). This definition of abstract Petri nets is closely related to other more complex notions of abstract Petri nets such as the definition of [Padberg, 1999]. It aims at defining the largest model in which the modelling of execution we will present and upon which our verification method is based can be applied. The definition of the structure and semantics of abstract Petri nets is analogous to the definition of the structure and semantics of Petri nets (Section 2.1.2 page 11) and coloured Petri nets (Section 2.1.5.2 page 19).

An abstract Petri net is defined within a framework where: $(\mathcal{D}, +_{\mathcal{D}})$ is a commutative monoid with its algebraic preordering $\leq_{\mathcal{D}}$. We denote $O_{\mathcal{D}}$ the minimal element of $\mathcal{D}$ and $\mathcal{L}(\mathcal{D}, \mathcal{D})$ the space of linear maps from $\mathcal{D}$ to $\mathcal{D}$.

Definition 41: Abstract Petri Net

An abstract Petri net AN is a tuple $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ where:

- P is a finite set of places,
- T is a finite set of transitions,
- $\mathcal{W}^- : P \times T \to \mathcal{L}(\mathcal{D}, \mathcal{D})$ is the pre-incidence function,
- $\mathcal{W}^+ : T \times P \to \mathcal{L}(\mathcal{D}, \mathcal{D})$ is the post-incidence function.

The state of an abstract Petri net is given by its marking, as proposed in the next definition.

Definition 42: Marking of an Abstract Petri Net

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract Petri net. A marking of AN is a total function $M : P \to \mathcal{D}$.

Two markings M_a and M_b are in relation $M_a \leq_{\mathcal{D}} M_b$ if and only if $\forall p \in P, M_a(p) \leq_{\mathcal{D}} M_b(p)$.

Let $g_P \in P, g_T \in T, G_P \subseteq P$ and $G_T \subseteq T$. We use the following notations:

- $g_P^\bullet = \{t | \mathcal{W}^+(g_P, t) \neq O_{\mathcal{D}}\}$ and ${}^\bullet g_P = \{t | \mathcal{W}^-(g_P, t) \neq O_{\mathcal{D}}\}$,
- $g_T^\bullet = \{p | \mathcal{W}^+(p, g_T) \neq O_{\mathcal{D}}\}$ and ${}^\bullet g_T = \{p | \mathcal{W}^-(p, g_T) \neq O_{\mathcal{D}}\}$,
- $G_P^\bullet = \cup_{g \in G_P} g^\bullet$, and ${}^\bullet G_P = \cup_{g \in G_P} {}^\bullet g$,
- $G_T^\bullet = \cup_{g \in G_T} g^\bullet$, and ${}^\bullet G_T = \cup_{g \in G_T} {}^\bullet g$.

The markings of an abstract Petri net evolve when transition instances are firing.

Definition 43: Transition Instance of an Abstract Petri Net

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract Petri net. A transition instance of AN is a couple (t, d), where $t \in T$ and $d \in \mathcal{D}$.

Let $t \in T$ and $d \in \mathcal{D}$, the transition instance (t, d) is *enabled* in a marking M if and only if:

$$\forall p \in {}^\bullet t, \mathcal{W}^-(p, t)(d) \leq_{\mathcal{D}} M(p).$$

When (t, d) is *enabled* in marking M_a, it may *fire*. If (t, d) *fires* in marking M_a, a new marking M_b is reached such that:

$$\forall p \in P, M_b(p) = M_a(p) -_{\mathcal{D}} \mathcal{W}^-(p, t)(d) +_{\mathcal{D}} \mathcal{W}^+(t, p)(d).$$

M_b is said to be *directly reachable* from M_a by the transition instance (t, d), written $M_a \xrightarrow{(t,d)} M_b$. The *reachability* relation is the reflexive and transitive closure of the *direct reachability* relation. Let $\sigma = (t_1, d_1), .., (t_n, d_n)$ be a sequence of transition instances, we denote by $M_a \xrightarrow{\sigma} M_b$ the fact that the marking M_b is reachable from the marking M_a by the sequence of transition instances σ.

Let $\mathcal{D}'$ be the Grothendieck group of $\mathcal{D}$ (i.e. the most general commutative group that arises from $\mathcal{D}$ by introducing additive inverses). Without loss of generality, let us suppose an abstract Petri net with no self loop, i.e. $\forall p, p^\bullet \cap {}^\bullet p = \emptyset$, where $P = \{p_1, .., p_n\}$ and $T = \{t_1, .., t_m\}$. This abstract Petri net can be represented by its corresponding incidence matrix $\mathcal{W} \in \mathcal{L}(\mathcal{D}, \mathcal{D}')^{(n,m)}$ where:

$$\mathcal{W}_{i,j} = \mathcal{W}^+(t_j, p_i) -_{\mathcal{D}'} \mathcal{W}^-(p_i, t_j).$$

Let M be a marking, it can be represented by a marking vector $\mathcal{M} \in \mathcal{D}^{(n,1)}$ where $\mathcal{M}_{i,1} = M(p_i)$. Likewise, let (t, d) be a transition instance, it can be represented by a transition instance vector $\mathcal{T} \in \mathcal{D}^{(m,1)}$ where:

$$\mathcal{T}_{j,1} = \begin{cases} d, & \text{if } t_j = t \\ O_{\mathcal{D}}, & \text{otherwise} \end{cases}$$

Let $\circledast$ denote the generalised matrix-multiplication where each product is replaced by a function composition (i.e $C = A \circledast B \Leftrightarrow C_{i,j} = \Sigma(A_{i,k} \circ B_{k,j})$). For $M_a \xrightarrow{(t,d)} M_b$, we have:

$$\mathcal{M}_b = \mathcal{M}_a + \mathcal{W} \circledast \mathcal{T}_{(t,d)}.$$

We denote $\mathcal{T}_{(t,d)}$ the transition instance vector representing the transition instance (t, d), and $\sigma \models \mathcal{T}$ the fact that $\mathcal{T} = \mathcal{T}_{(t_1,d_1)} +_{\mathcal{D}} .. +_{\mathcal{D}} \mathcal{T}_{(t_n,d_n)}$.

In the context of abstract Petri nets, we define abstract workflow nets, a generalisation of workflow Petri nets (Section 2.1.6 page 21).

Definition 44: Abstract Workflow Net

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract Petri net. The abstract Petri net AN is an abstract workflow net if and only if:

- AN has two special places i and o where ${}^\bullet i = \emptyset$ and $o^\bullet = \emptyset$, and
- for each node $n \in (P \cup T)$ there exists a path from i to o passing through n.

We denote $\mathcal{M}_i$ the set of initial markings of an abstract workflow net where:

$$M_i \in \mathcal{M}_i \Leftrightarrow M_i(i) > O_{\mathcal{D}} \wedge \forall\ p \in P \setminus \{i\}, M_i(p) = O_{\mathcal{D}}.$$

Likewise we denote $\mathcal{M}_o$ the set of final markings of an abstract workflow net where:

$$M_o \in \mathcal{M}_o \Leftrightarrow M_o(o) > O_{\mathcal{D}} \wedge \forall p \in P \setminus \{o\}, M_o(p) = O_{\mathcal{D}}.$$

A *correct execution* of an abstract workflow net is an execution $M_i \xrightarrow{\sigma} M_o$, where $M_i \in \mathcal{M}_i$ and $M_o \in \mathcal{M}_o$.

The behaviour of AN is defined as the set Σ of all its *correct executions*.

Next, we proceed to explicitly show that Petri nets (Section 2.1.2 page 11), and coloured Petri Net nets (Section 2.1.5.2 page 19) are abstract Petri nets. Furthermore, we introduce weighted transitions abstract workflow nets, an extension of abstract workflow nets, that associates weights to transition instances in order to enable performance analysis.

Petri Nets as Abstract Petri Nets

A ordinary/generalised Petri net (Section 2.1.2 page 11) is an abstract Petri net defined within the framework where $\mathcal{D}$ is the set $\mathbb{N}$ of natural numbers, $+_{\mathcal{D}}$ is the standard *addition* operation over $\mathbb{N}$, and $\leq_{\mathcal{D}}$ is the standard *less-than-or-equal* relation over $\mathbb{N}$.

Let $N = \langle P, T, F, W \rangle$ be a generalised Petri net, N is an abstract Petri net $\langle aP, aT, \mathcal{W}^-, \mathcal{W}^+ \rangle$ where:

- the set of place P and aP are the same ($P = aP$),
- the set of transition T and aT are the same ($T = aT$),
- the pre-incidence function is defined by $\mathcal{W}^-(p, t)(x) = W(p, t) * x$ if $(p, t) \in (F \cap (P \times T)$ and $\mathcal{W}^-(p, t)(x) = O$ otherwise,

– the post-incidence function is defined by $\mathcal{W}^+(t,p)(x) = W(t,p) * x$ if $(t,p) \in (F \cap (T \times P)$ and $\mathcal{W}^+(t,p)(x) = O$ otherwise,

– a transition instance (t,d) is seen as the transition t occurring d times.

Example 18: Illustration of a generalised Petri net as an abstract Petri net

Consider the generalised Petri net of Example 2 page 13 depicted in Figure 2.2(a) page 13. This generalised Petri net is composed of three places (H_2O, O_2 and H_2) and a transition (t_1). Its incidence matrix is defined as: $[-2,1,2]^\top$. The abstract Petri net corresponding and equivalent to this generalised Petri net is composed of the same three places and transition, and its incidence matrix is defined as: $[x \mapsto -2*x, x \mapsto 1*x, x \mapsto 2*x]^\top$.

Coloured Petri Nets as Abstract Petri Nets

A coloured Petri net (Section 2.1.5.2 page 19) is an abstract Petri net defined within the framework where $\mathcal{D}$ is the set $C_1 \times .. \times C_u$ over the colours $C_1, .., C_u$ of the coloured Petri net , $+_{\mathcal{D}}$ is the element-wise *addition* operation over u-uples, and $\leq_{\mathcal{D}}$ is the element-wise *less-than-or-equal* relation over u-uples. Let π_{C_i} be the projection of C_i from $C_1 \times .. \times C_u$ and Π_{C_i} be the projection of $C_1 \times .. \times C_u$ from C_i.

Let $N = \langle (P,T,C,W) \rangle$ be a coloured Petri net, N is an abstract Petri net $\langle aP, aT, \mathcal{W}^-, \mathcal{W}^+ \rangle$ where:

– the set of place P and aP are the same ($P = aP$),

– the set of transition T and aT are the same ($T = aT$),

– the pre-incidence function is defined by $\mathcal{W}^-(p,t) = \Pi_{C(p)} \circ W(t,p) \circ \pi_{C(p)}$,

– the post-incidence function is defined by $\mathcal{W}^+(t,p) = \Pi_{C(p)} \circ W(t,p) \circ \pi_{C(p)}$,

– a transition instance (t,d) is then seen as the transition t associated with the binding $\pi_{C(t)}(d)$.

Example 19: Illustration of a coloured Petri net as an abstract Petri net

Consider the coloured Petri net of Example 4 page 20 depicted in Figure 2.8 page 20. This coloured Petri net have two colors (C_1 and C_2) and is composed of two places (p_1, and p_2) and a transition (t_1) such that $C(p_1) = C(t_1) = C_1$ and $C(p_2) = C_2$. Its incidence matrix is defined as: $[[v1,v2,v3] \mapsto [-v1,-v2,-v3], [v1,v2,v3] \mapsto [v1,v2,v3]]^\top$. The abstract Petri net corresponding and equivalent to this coloured Petri net is composed of the same three places and transition, and its incidence matrix is defined as: $[([v1,v2,v3],[0,0,0]) \mapsto ([-v1,-v2,-v3],[0,0,0]), ([v1,v2,v3],[0,0,0]) \mapsto ([0,0,0],[v1,v2,v3])]^\top$.

Weighted Transition Abstract Workflow Nets

Analogously to weighted transition Petri nets (Section 2.1.5.3 page 21), weighted transitions abstract workflow nets are an extension of abstract workflow nets. They associate weights to transition instances in order to be able to perform performance analysis.

They rely upon the definition of a weight function defined within a framework where $(C, +_C)$ is a commutative monoid. We denote O_C the minimal element of C.

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net, a weight function of AN is a total function $C : T \times \mathcal{D} \to C$ assigning a weight to each transition instances.

Let $AN_w = \langle AN, C \rangle$ be the weighted transition abstract workflow net formed from the abstract workflow net AN associated with its weight function C.

The weight of a transition instance (t, d) is then given by $C(t, d)$. Let M_a, M_b be marking of AN, and $\mathcal{T}$ a transition instance sequence such that $M_a \xrightarrow{\mathcal{T}} M_b$, the weight associated to execution $\mathcal{T}$ is given by the execution weigh function $\mathfrak{C}$ defined as $\mathfrak{C}(\mathcal{T}) = \sum_{t \in T} \mathcal{T}(t) * C(t)$.

We defined abstract workflow nets and presented their relation to workflow nets, and coloured workflow nets. The next section focuses on the definition of extended modal specifications over abstract workflow nets.

3.2.2/ Extended Abstract Modal Specification

This section introduces extended modal specification over abstract workflows net.

It generalises the concept of extended modal specifications over ordinary workflow nets to the realm of abstract workflow nets. The main additions are that modal properties, beside considering constraints over the transitions involved in correct executions and their relations, consider constraints on the initial markings, the final markings, as well as the transition instances.

To this end, the language $S(AN)$ of well-formed modal specification formulae associated with an abstract workflow net $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ is defined as follows.

Let be $e \in \mathcal{D}^{(l,c)}$ where $l, c \in \mathbb{N}$, we denote $F(e, l, c)$ the set of formulae defined by the following grammar of axiom A, where $i \in 1..l$, $j \in 1..c$, and G is the axiom of the grammar of elements of $\mathcal{D}$:

- $A \to (A \wedge A) | (A \vee A) | (\neg A) | B$
- $B \to C \leq_{\mathcal{D}} C$
- $C \to D +_{\mathcal{D}} D$
- $D \to e_{i,j} | G$

Given $m \in F(e, l, c)$, we denote $e \models m$ the fact that e satisfies m, where $\wedge$, $\vee$, $\neg$, and $\leq_{\mathcal{D}}$ assume standard semantics. Using this, a marking vector $\mathcal{M}$ can be constrained by a formula $m \in F(\mathcal{M}, n, 1)$, and a transition vector $\mathcal{T}$ can be constrained by a formula $m' \in F(\mathcal{T}, m, 1)$.

Definition 45: Well-formed Abstract Modal Specification Formulae

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net. The language $S(AN)$ of well-formed *modal* specification formulae of AN is defined by the following grammar of axiom A:

- $A \to (A \wedge A) | (A \vee A) | (\neg A) | B$
- $B \to t[e]$ where $t \in T$ and $e \in F(d, 1, 1)$ with d a variable of domain $\mathcal{D}$.

These well-formed abstract modal specification formulae allow specifiers to describe abstract workflow nets behaviour based on the transition instances involved.

Formally, let $m \in S(AN)$ be a well-formed abstract modal specification formula and $\mathcal{T}$ the transition instance vector of a correct execution of AN, we denote $\mathcal{T} \models m$ the fact that the behaviour expressed by m is satisfied by the transition instance vector $\mathcal{T}$. Formally, given $t[e]$ where $t \in T$ and $e \in F(d, 1, 1)$ with d is a variable of domain $\mathcal{D}$, we have $\mathcal{T} \models t[e] \Leftrightarrow \mathcal{T}_{i,1} \models e[d/\mathcal{T}_{i,1}]$ where $t_i = t$ and $e[d/\mathcal{T}_{i,1}]$ is the formula e, where all occurrences of d are replaced by $\mathcal{T}_{i,1}$. Further, given $A_1, A_2 \in S(AN)$, we have $\mathcal{T} \models (A_1 \wedge A_2) \Leftrightarrow \mathcal{T} \models A_1 \wedge \mathcal{T} \models A_2, \mathcal{T} \models (A_1 \vee A_2) \Leftrightarrow \mathcal{T} \models A_1 \vee \mathcal{T} \models A_2$, and $\mathcal{T} \models (\neg A_1) \Leftrightarrow \neg(\sigma \models A_1)$.

Next, we extend well-formed abstract modal specification formulae in order to consider constraints on the initial markings as well as the final markings.

To this end, we define the set of well-formed extended abstract modal specification formulae as follows.

Definition 46: Extended Abstract Modal Specification Formulae

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net. The set of well-formed extended abstract modal specification formulae of AN, noted $S_e(AN)$, is defined by the set of tuple $\langle cM_i, cM_o, m \rangle$ where:

- $cM_i \in F(M_i, m, 1)$ are the constraints on the initial marking,
- $cM_o \in F(M_o, m, 1)$ are the constraints on the final marking,
- $m \in S(AN)$ is the constraint on the transition instances and their relations defined by a well-formed abstract modal specification formula.

These well-formed extended abstract modal specification formulae allow specifiers to express abstract modal properties about abstract workflow nets correct executions.

Formally, let $M_i \in \mathcal{M}_i$ be an initial marking of AN, $M_o \in \mathcal{M}_o$ be a final marking of AN, σ be a correct execution of AN such that $M_i \xrightarrow{\sigma} M_o$, and $\mathcal{T}$ be a transition instance vector such that $\sigma \models \mathcal{T}$, we say that a well-formed extended abstract modal specification formula $m = \langle cM_i, cM_o, m \rangle$ is satisfied by the correct execution σ if and only if $\mathcal{T} \models m \wedge M_i \models cM_i \wedge M_o \models cM_o$.

Every modal property expressed by a well-formed extended abstract modal specification formula can be interpreted as a *may*-formula (resp. *must*-formula), describing a behaviour that has to be ensured by at least one (resp. all) correct execution(s) of the specified abstract workflow net.

Formally, an abstract workflow AN satisfies a *may*-formula $f_{may} = \langle cM_{i-may}, cM_{o-may}, m_{may} \rangle$, noted $AN \models_{may} f_{may}$, if and only if $\exists\, \sigma \in \Sigma, M_i \in \mathcal{M}_i, M_o \in \mathcal{M}_o, M_i \xrightarrow{\sigma} M_o \wedge \sigma \models \mathcal{T} \wedge \mathcal{T} \models m_{may} \wedge M_i \models cM_{i-may} \wedge M_o \models cM_{o-may}$. Further, an abstract workflow AN satisfies a *must*-formula $f_{must} = \langle cM_{i-must}, cM_{o-must}, m_{must} \rangle$, noted $AN \models_{must} f_{must}$, if and only if $\forall\, \sigma \in \Sigma, M_i \in \mathcal{M}_i, M_o \in \mathcal{M}_o, M_i \xrightarrow{\sigma} M_o \wedge \sigma \models \mathcal{T} \wedge \mathcal{T} \models m_{may} \wedge M_i \models cM_{i-may} \wedge M_o \models cM_{o-may}$.

Analogously to the way extended modal specifications are defined for ordinary workflow nets, we define abstract extended modal specifications for abstract workflow nets as follows.

Definition 47: Abstract Extended Modal Specification

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net. An abstract extended modal specification of AN is tuple $\langle a_{must}, A_{may} \rangle$ where:

- a_{must} is a *must*-formula, and
- A_{may} is a set of *may*-formulae.

Given an abstract workflow net AN, we say that AN satisfies (i.e. is conform to) an abstract extended modal specification $\langle a_{must}, A_{may} \rangle$, noted $AN \models \langle a_{must}, A_{may} \rangle$, if and only if:

$$N \models_{must} a_{must}, \text{ and}$$

$$\forall\, a_{may} \in A_{may}, N \models_{may} a_{may}.$$

Note that in the case where AN is an abstract workflow nets extended with performance indicators (i.e. AN is a weighted transition abstract workflow net, Section 3.2.1 page 58), abstract extended modal specification can be further extended by considering constraints on the total weight of correct executions.

To this end, a *may*-formula $f_{may} = \langle cM_{i-may}, cM_{o-may}, m_{may} \rangle$ (resp. a *must*-formula $f_{must} = \langle cM_{i-must}, cM_{o-must}, m_{must} \rangle$ is extended by a constraint $cC \in F(w, m, 1)$ where w is the execution weight associated with the modelled executions.

It follows that a *may*-formula f_{may} (resp. a *must*-formula f_{must}) extended by considering a constraint cC on the total weights of correct executions is satisfied by a weighted transitions abstract workflow net AN if and only if there exists a correct execution satisfying f_{may} (resp. all correct executions satisfy f_{must}) such that its (resp. their) total weight satisfies cC.

3.2.3/ Modelling Executions of Abstract Workflow Nets

This section aims at modelling correct executions of abstract Petri net (Section 3.2.1 page 55) by a constraint system which is then solved to validate or invalidate some properties of interest. This modelling is similar to the modelling previously proposed to model correct executions of ordinary workflow nets.

We begin with the generalisation of the state equation to the domain of abstract workflow nets.

Theorem 18: State equation

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net. If a marking M_b is reachable from M_a by the transition instances sequence $\sigma = (t_1, d_1), .., (t_n, d_n)$ (i.e. $M_a \xrightarrow{\sigma} M_b$) then: $\mathcal{M}_b = \mathcal{M}_a + \mathcal{W} \circledast \mathcal{T}$ where $\sigma \models \mathcal{T}$.

Proof. By definition we have $\mathcal{M}_b = \mathcal{M}_a + _{\mathcal{D}} (\mathcal{W} \circledast \mathcal{T}_{(t_1,d_1)}) + _{\mathcal{D}} .. + _{\mathcal{D}} (\mathcal{W} \circledast \mathcal{T}_{(t_n,d_n)})$ which is equivalent to $\mathcal{M}_b = \mathcal{M}_a + _{\mathcal{D}} \mathcal{W} \circledast (\mathcal{T}_{(t_1,d_1)} + _{\mathcal{D}} .. + _{\mathcal{D}} \mathcal{T}_{(t_n,d_n)})$ as $\circledast$ is distributive. □

We denote the constraint system of Theorem 18 by $\mathcal{S}_a(M_a, M_b)$. This constraint system defines an over approximation of the executions reaching M_b from M_a. Indeed, analogously to $\mathcal{S}$ defined over workflow nets, $\mathcal{S}_a$ does not consider the order in which transitions are fired, which leads to spurious solutions.

We continue and refine solutions of $\mathcal{S}_a$ using structural features of their associated subnets composed of places (except places only marked in the initial marking, and places only marked in the final marking), transitions, and arcs involved in the solution of $\mathcal{S}_a$.

As we are only interested in the structure of such subnets, it is sufficient to build them as ordinary Petri nets, i.e. using arc weight less or equal to 1.

Definition 48: Subnet of a solution of $\mathcal{S}_a$

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net, M_a, M_b be two markings of AN and $\mathcal{T}$ be a solution of the constraint system $\mathcal{S}_a(M_a, M_b)$ of AN. Let $\mathcal{E}^+ = \mathcal{W}^+ \circledast \mathcal{T}$ and $\mathcal{E}^- = \mathcal{W}^- \circledast \mathcal{T}$. We define the subnet $sAN(\mathcal{T}) = \langle sP, sT, sF \rangle$ an ordinary Petri net where:

- $sP = \{p \in P \mid (O_{\mathcal{D}} <_{\mathcal{D}} \mathcal{E}^+(p) \wedge M_a(p) <_{\mathcal{D}} \mathcal{E}^+(p)) \vee (O_{\mathcal{D}} <_{\mathcal{D}} \mathcal{E}^-(p) \wedge M_b(p) <_{\mathcal{D}} \mathcal{E}^-(p))\}$
- $sT = \{t \in T \mid O_{\mathcal{D}} <_{\mathcal{D}} \mathcal{T}(t)\}$
- $sF = \{(p, t) \mid p \in sP \wedge t \in sT)\} \cup \{(t, p) \mid t \in sT \wedge p \in sP)\}$

We know from the previous section that if the subnet of a solution to $\mathcal{S}_a$ contains a siphon or a trap then this solution is a spurious solution. We also know that the subnet of a solution to $\mathcal{S}_a$ contains a siphon (resp. a trap) if and only if it is also a trap (resp. a siphon). Using these rules to refine $\mathcal{S}_a$ eliminates a large number of its spurious solution.

Definition 49: State Equation + Absence of Siphon Constraint System

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net, and M_a, M_b be two markings of AN, the constraint system $\mathcal{Q}_a(AN, M_a, M_b)$ associated with it is:

- $\mathcal{S}(N, M_a, M_b) \models \mathcal{T}$
- $\nexists\, \xi$ such that $\mathcal{B}(sAN(\nu)) \models \xi$

where $\mathcal{T} : T \to \mathcal{D}$ is a valuation function.

However, the set of solutions of $\mathcal{S}_a(M_a, M_b)$, whose subnets contain no siphon, equivalently no trap or no P-semiflow (i.e. solutions of $\mathcal{Q}_a(M_a, M_b)$), still defines an over approximation of the executions reaching M_b from M_a.

To further refine the solutions $\mathcal{Q}_a$, we need to restrict the structure of their subnets. We propose to restrict them to the class of marked graphs (Section 2.1.4 page 16). This structural restriction is sufficient to guarantee the existence of at least an execution for every solution.

Definition 50: State Equation + Absence of Siphon Constraint + MG System

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net, and M_a, M_b be two markings of AN the constraint system $\mathcal{D}_a(AN, M_a, M_b)$, associated with it, is:

- $\mathcal{Q}(N, M_a, M_b) \models \mathcal{T}$
- $sAN(\mathcal{T}) = \langle sP, sT, sF \rangle$ is a marked graph (i.e. $\forall p \in sP, |^\bullet p| \leq 1 \wedge |p^\bullet| \leq 1$)

where $\mathcal{T} : T \to \mathcal{D}$ is a valuation function.

The solutions of the constraint system of Definition 50 are related to the execution of an abstract workflow net AN as stated by Theorem 19.

Theorem 19: Abstract Path Existence

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net, and M_a, M_b be two markings of AN. If there exists a valuation $\mathcal{T} : T \to \mathcal{D}$ such that $\mathcal{D}_a(N, M_a, M_b) \models \mathcal{T}$ then there exists a transition instance sequence σ such that $M_a \xrightarrow{\sigma} M_b$ and $\sigma \models \mathcal{T}$.

Proof. Suppose there exists a valuation $\mathcal{T} : T \to \mathcal{D}$ such that $\mathcal{D}_a(N, M_a, M_b) \models \mathcal{T}$. By definition of $\mathcal{D}_a(N, M_a, M_b)$ we have $sN(\mathcal{T})$ is a marked graph with no siphons nor traps. It follows from Theorem 5 page 34 that $sN(\mathcal{T})$ is a live marked graph from any marking (in particular from an initial marking $M_\emptyset$ such that $\forall\, p \in sP, M_\emptyset(p) = 0$). Therefore $\forall\, p \in sP$ there exists $t_{pre} \in {}^\bullet p, t_{post} \in p^\bullet$ such that $\forall\, \sigma, M_\emptyset \xrightarrow{\sigma} M_\emptyset$ is an execution of $sN(\mathcal{T})$, we have $O_{t_{post}}(\sigma) = 1 \Rightarrow O_{t_{pre}}(\sigma) = 1$. By induction it follows that there exists σ such that $M_\emptyset \xrightarrow{\sigma} M_\emptyset$ an execution of $sN(\nu)$ where $\forall t \in T$ such that $\mathcal{T}(t) > O_\mathcal{D}$ we have $O_t(\sigma) = 1$. As data consistency is ensured by the state equation, it follows that there exists $\sigma*$, a transition instance sequence obtained by replacing every occurrence of any transition $t \in sT$ by its corresponding transition instance $\mathcal{T}(t)$, such that $M_a \xrightarrow{\sigma*} M_b$ is an execution of AN where $\sigma* \models \mathcal{T}$. □

It follows that the set of solutions of $\mathcal{D}_a(AN, M_a, M_b)$ defines an under approximation of the executions of AN reaching M_b from M_a.

We call a segment any executions modelled by $\mathcal{D}_a$ and proceed to model any executions as the composition of segments.

Definition 51: k-segment Abstract Execution

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net, M_a, M_b be two markings of AN and $k \in \mathbb{N}$, the constraint system $\mathcal{U}_a(AN, M_a, M_b, k)$, associated with it, is:

- $\exists\, M_1, \nu_1, \mathcal{D}_a(AN, M_a, M_1) \models \mathcal{T}_1$
- $\forall\, i \in \{2, .., k-1\}, \exists\, M_i, \mathcal{T}_i, \mathcal{D}_a(N, M_{i-1}, M_i) \models \mathcal{T}_i$
- $\exists\, \mathcal{T}_k, \mathcal{D}_a(N, M_{k-1}, M_b) \models \mathcal{T}_k$
- $\forall\, n \in T, \mathcal{T}(n) = \sum_{i \in \{1,..,k\}} \mathcal{T}_i(n)$

where $\mathcal{T} : T \to \mathcal{D}$ is a valuation function.

The solutions of the constraint system of Definition 51 are related to the execution of an abstract workflow net AN as stated by Theorem 20.

Theorem 20: Abstract Path Existence

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+ \rangle$ be an abstract workflow net, M_a, M_b be two markings of AN. There exists a transition instance sequence σ such that $M_a \xrightarrow{\sigma} M_b$ if and only if there exist $k \in \mathbb{N}$ a number of segments and $\mathcal{T} : T \to \mathcal{D}$ a valuation function such that $\mathcal{U}_a(N, M_a, M_b, k) \models \mathcal{T}$ and $\sigma \models \mathcal{T}$.

Proof. ($\Rightarrow$) Suppose there exists a transition instance sequence $\sigma = t_1, .., t_k$ such that $M_a \xrightarrow{\sigma} M_b$.

By definition there exists $M_1, .., M_{k-1}$ such that $M_a \xrightarrow{t_1} M_1 \xrightarrow{t_2} .. \xrightarrow{t_{k-1}} M_{k-1} \xrightarrow{t_k} M_b$. As the firing of a single transition instance t can be modelled by a valuation $\mathcal{T}_t : T \to \mathcal{D}$ satisfying $\mathcal{S}_a$ and the associated subnet $sN(\mathcal{T}_t)$ is a marked graph without siphons composed of only the transition t, we can infer that $\mathcal{T}_t$ is also a valuation satisfying $\mathcal{D}_a$. It follows that there exist $\mathcal{T}_1, .., \mathcal{T}_k$ such that $\mathcal{D}(N, M_a, M_1) \models \mathcal{T}_1, .., (N, M_{k-1}, M_b) \models \mathcal{T}_k$. Let $\mathcal{T} : T \to \mathbb{N}$ be a valuation function such that $\forall\, p \in T, \nu(p) = \sum_{i\in\{1,..,k\}} \mathcal{T}_i(p)$. By definition of $\mathcal{U}_a$, we can conclude that $\mathcal{U}_a(AN, M_a, M_b, k) \models \mathcal{T}$ and $\sigma \models \mathcal{T}$.

($\Leftarrow$) Follows from Theorem 19. □

The constraint system $\mathcal{U}$ enables the modelling of finite executions of abstract workflow nets by considering their initial and final markings, and their sequences of transition instances.

This section presented a modelling framework which enables the modelling of abstract workflow nets through the construction of constraint systems. The next section takes advantage of this modelling to verify (extended) abstract modal specifications.

3.2.4/ Verifying Extended Abstract Modal Specifications

Analogously to the previous section, this section describes how the modelling of correct executions of an abstract workflow net can be used to verify the validity of an abstract extended modal specifications.

To this end, the following constraint system uses the constraint system $\mathcal{U}_a$ to model only correct executions of AN composed of $k \in \mathbb{N}$ segments satisfying a given abstract modal property.

Definition 52: k-segment Abstract Execution

Let $AN = \langle P, T, \mathcal{W}^-, \mathcal{W}^+\rangle$ be an abstract workflow net, $k \in \mathbb{N}$, and $f = \langle cM_i, cM_o, m\rangle$ be an abstract modal formula of AN, the constraint system $\mathcal{V}_a(AN, k, f)$ associated with it is:

- $M_i \models cM_i$
- $M_o \models cM_o$
- $\mathcal{U}_a(AN, M_i, M_o, k) \models \mathcal{T}$
- $\mathcal{T} \models m$

where $(M_i, M_o, \mathcal{T}) : (P \to \mathcal{D}, P \to \mathcal{D}, T \to \mathcal{D})$ is a valuation function.

It follows that there exists a correct execution, composed of at most $k \in \mathbb{N}$ segments, of AN, an abstract workflow net, such that it satisfies an abstract modal formula f if and only if the constraint system $\mathcal{V}_a(AN, k, f)$ is satisfiable.

Let $f = \langle cM_i, cM_o, m\rangle$ be an abstract modal formula of an abstract workflow net AN. To verify the *K-bounded* validity of f interpreted as a *may*-formula, it is sufficient to determine the existence of a correct execution modelled by K segments with an initial marking satisfying cM_i, a final marking satisfying cM_o and with the behaviour satisfying m. Therefore, a *may*-formula f is *K-bounded* valid with respect to AN if and only if $\mathcal{V}_a(AN, k, f)$ is satisfiable.

Similarly, determining the *K-bounded* validity of f interpreted as a *must*-property, can be done by determining the non-existence of a correct execution modelled by K segments with an ini-

tial marking satisfying cM_i, a final marking satisfying cM_o where the behaviour of $\neg(m)$ is satisfied. Therefore, a *must*-formula f is *K-bounded* valid with respect to AN if and only if $\mathcal{V}_a(AN, k, \langle cM_i, cM_o, \neg m\rangle)$ is not satisfiable.

Finally, in the case where AN is an abstract extended with performance indicators (i.e. AN is a weighted transitions abstract Petri net, Section 2.1.5.3 page 21), we need to consider the constraint cC given over the total execution weight. To this end, note that the total weight of an execution modelled by $\mathcal{U}_a(N, M_a, M_b, k) \models \mathcal{T}$ is given by $\mathfrak{C}(\mathcal{T}) = \sum_{t\in T} \mathcal{T}(t) * C(t)$. Therefore it is necessary to add the following constraint to $\mathcal{U}_a(N, M_a, M_b, k)$: $\mathfrak{C}(\mathcal{T}) \models cC$.

3.3/ Synthesis

This chapter presented an approach to the verification of modal specifications.

It defined extended modal specifications, an extension of modal specifications that enables the definition of modal behaviour involving several transitions.

An innovative modelling framework of workflow nets (resp. abstract workflow nets) executions has also been defined. This modelling framework is based on the definition of several constraint systems. More precisely, it defines constraint systems whose solution space over-approximate the set of correct executions of workflow nets (resp. abstract workflow nets). Each of this over-approximation is built by refining the previous one starting from a well-known approximation: the state equation. Further, it also defined a constraint system whose solution space under-approximates the set of correct executions of workflow nets (resp. abstract workflow nets). This latter constraint system is then used to define segments of execution (i.e. partial execution structurally verifiable). It then proceeded to demonstrate that the concatenation of such segments could be used to model any correct executions of workflow nets (resp. abstract workflow nets).

One of the advantages of this modelling is that the search is quite focussed from the beginning as we traverse the solution space of the state equation and its refinements rather than the underlying semantic labelled transition systems of workflow nets. Another advantage of such modelling is that only the segments ordering is computed, the transitions ordering within segments is not. Therefore, this modelling is adapted to the verification of properties where only the presence or absence of transitions is considered (e.g., extended modal specifications).

Finally, it presented how such modelling framework is used to verify the (in)validity of extended modal specifications over workflow nets (resp. abstract workflow nets). This verification approach, based upon the previously presented modelling may perform well when verifying the conformity of workflow nets with respect to valid *must*-formulae as well as invalid *may*-formulae, thanks to adequate use of over-approximations. Furthermore, this extended modal specification verification approach requires the use of constraint systems satisfiability checks which can be handled by third party constraint solvers. This allows the proposed verification method to benefit from existing mature and efficient constraint solvers.

In order to enhance verification approaches such as the one presented in this chapter, the following chapter presents powerful reduction methods preserving properties of interest such as generalised soundness and correctness of a given modal specification. It also presents how these reduction methods can be employed as pre-processing steps to reduce workflow nets size in order to verify the preserved properties on smaller instances.

4

Reduction methods

> "All problems in computer science can be solved by another level of indirection."
>
> — David Wheeler

Contents

4.1 Φ^*: A workflow nets reduction kit 68

4.2 Semi-Decision of Generalised Soundness 77

4.3 Preprocessing Modal Specification Verification 79

4.3.1 Reduction based on hierarchical workflow nets 80

4.3.2 Reduction based on reduction rules 82

4.4 Synthesis 86

The development of large and intricate workflow nets can be a difficult task which requires powerful structuring mechanisms [Dittrich, 1989]. It also forces modellers to follow strict abstraction patterns in order to produce quality workflow nets [van der Aalst et al., 2000]. To cope with these development difficulties, stepwise refinement (Section 2.1.9 page 25) is often used to ensure reliability and ease verification.

Verification of workflow nets is an a posteriori approach: given a workflow net, it checks whether properties (e.g., generalised soundness, extended modal specification) hold. Although for workflow nets these properties are known to be decidable [Esparza, 1998, van der Aalst et al., 2011], their verification is a very time consuming task due to its high complexity (EXPSPACE) with respect to the size of the workflow net under analysis [Lipton, 1976].

Unfortunately, most often, the abstraction mechanisms, used by modellers of workflow nets, are not explicitly given beside the clear advantages they bring to their analysis.

As seen in Section 2.2.3 page 35, reduction rules have the ability to reduce workflow nets size while strongly preserving properties of interest. This allows the analysis of studied properties to be performed on reduced workflow nets, in many cases, greatly decreasing its complexity by alleviating state explosion of their state space, which undermines state exploration methods.

More generally, reduction rules are abstraction operations: they reduce the level of details of workflow nets. They aim at capturing the abstraction mechanisms used by modellers of workflow nets. It follows that the inversion of reduction rules (i.e. synthesis rules) are refinement operations. Conceptually, this leads to an analysis paradigm where the analysis of workflow nets is substituted by the analysis of their constructions.

The first section of this chapter presents reduction rules strongly preserving generalised soundness over workflows nets, an essential and necessary correctness property that must be satisfied by workflow nets.

Afterwards, the second section describes how these reduction rules can be used to efficiently semi-decide generalised soundness.

The third section introduces pre-processing steps reducing workflow nets size so that the analysis can be carried out on smaller instances. More specifically, it highlights how reduction methods can be applied as pre-processing steps to the verification of extended modal specifications over workflow nets.

Finally, the last section concludes this chapter.

4.1/ Φ^*: A WORKFLOW NETS REDUCTION KIT

This section defines Φ^*, a kit of reduction rules over workflow nets that strongly preserve generalised soundness (Definition 24 page 25).

In Section 2.2.2.4 page 34 we saw that the k-soundness of a workflow net is equivalent to the liveness and boundedness of its k-closure. Therefore, known reduction rules (Section 2.2.3.2 page 36) which strongly preserve liveness and boundedness of Petri nets also strongly preserve generalised soundness of workflow nets. Conversely, the reduction rules introduced in this section strongly preserve liveness and boundedness.

The reduction kit defined in this section generalises and adapts previously known reduction rules (Section 2.2.3.2 page 36) to the realm of workflow nets. They extend the range of workflow nets reducible in such a way.

The presented reduction rules are applicable to arbitrary workflow nets and are not restricted to subclasses. Further, their *conditions of application* are defined solely on the structure of the considered workflow nets. This allows their application to be performed statically in an efficient manner. Also, note that each rule's conditions of application describe an infinity of workflow patterns (i.e. structural configurations) to which they can be applied.

In what follows, each workflow net reduction rule is defined by describing the conditions of application under which it can be applied to a source workflow net $N = \langle P, T, F \rangle$, and the *construction algorithm* which is applied to N to produce a target workflow net $\tilde{N} = \langle \tilde{P}, \tilde{T}, \tilde{F} \rangle$.

For clarity, every rule is illustrated by figures depicting two of the possible applications of this rule. The first one only considers the plain element of the figure and corresponds to the minimal pattern. The second one considers both the mandatory (plain) and the optional (dashed) elements, and corresponds to a possible extended pattern. In this way, figures are not exhaustive but aim to ease the understanding of the rules that are formally described by the related conditions of application and the construction algorithms.

There are a total of six rules presented in the following subsections: one removing a place (R_1), two removing a transition (R_2, R_3), two removing a place and a transition (R_4, R_5), and finally one removing a strongly connected component: (R_6).

R_1: Remove Place

We define $\phi_{RemoveP}$, a workflow net reduction rule, which strongly preserves generalised soundness and consists in removing a place for which there exists a set of places with the same input transitions as well as the same output transitions. Places removed in such a way are called redundant places as they do not modify the set of correct executions of a workflow net.

This rule generalises the *Fusion of Parallel Places* rule described in [Murata, 1989] and the *Abstraction of Parallel Places* rule described in [Hichami et al., 2014]. This rule can also be viewed as an adaptation of the rule ϕ_S of [Desel et al., 2005] (i.e. one of the three reduction rules proved to be complete with respect to the subclass of free-choice Petri nets) to the context of ordinary workflow nets.

The inverse of this rule is the only synthesis rule able to add a single place to a workflow net while preserving generalised soundness and is notably used to introduce concurrency. In the context of Petri net, a self-loop place (i.e. a place p such that ${}^\bullet p = p^\bullet$) can be added without compromising liveness and boundedness. However, this requires changing the initial marking which is not possible in the context of workflow nets. Nonetheless, a generalisation of this rule could be applied to an extension of workflow nets modelling resources by marked places.

Figure 4.1 illustrates the reduction rule $\phi_{RemoveP}$ formally described as follows.

Conditions on N:	Construction of $\tilde{N}$:
– $\exists\, p \in P \setminus \{i, o\}$	– $p^\bullet = \{ot_1, .., ot_n\}$
– $\exists\, G = \{g_1, .., g_n\} \subseteq P \setminus \{i, o, p\}$	– $outArc = \{p\} \times p^\bullet$
– $p^\bullet = G^\bullet$	– ${}^\bullet p = \{it_1, .., it_m\}$
– ${}^\bullet p = {}^\bullet G$	– $inArc = {}^\bullet p \times \{p\}$
– $\forall i, j \in \{1, .., n\}, i \neq j \Rightarrow {}^\bullet g_i \cap {}^\bullet g_j = g_i^\bullet \cap g_j^\bullet = \emptyset$	– $\tilde{P} = P \setminus \{p\}$
	– $\tilde{T} = T$
	– $\tilde{F} = F \setminus (inArc \cup outArc)$

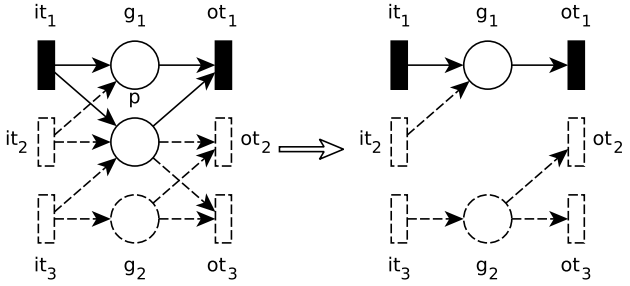


Figure 4.1: Reduction rule $\phi_{RemoveP}$ (R_1)

The soundness of $\phi_{RemoveP}$, with respect to generalised soundness, is given by the following theorem.

Theorem 21: Soundness of $\phi_{RemoveP}$

$\phi_{RemoveP}$ is a workflow net reduction rule which strongly preserves generalised soundness.

Proof. (Sketch). Let $f : (\tilde{P} \to \mathbb{N}) \to (P \to \mathbb{N})$ be a bijective function such that $f(M)(g) = M(g)$ for all $g \in \tilde{P}$ and $f(M)(p) = M(g_1) + .. + M(g_n)$. By conditions on N, every transition that produces (resp. consumes) a token in any of the places of G also produces (resp. consumes) a token in p. Consequently, $\forall k \in \mathbb{N}$ one has: $M \in \mathcal{R}^{\tilde{N}}(M^{\tilde{N}}_{i(k)}), M^{\tilde{N}}_o(k) \in \mathcal{R}^{\tilde{N}}(M) \Leftrightarrow f(M) \in \mathcal{R}^N(M^N_{i(k)}), M^N_o(k) \in \mathcal{R}^N(f(M))$, and transitions $ot_1, .., ot_n, it_1, .., it_m$ of N are not dead if and only if transitions $ot_1, .., ot_n, it_1, .., it_m$ of $\tilde{N}$ are not dead. □

R_2: Remove Transition

We define $\phi_{RemoveT}$, a workflow net reduction rule, which strongly preserves generalised soundness and consists in removing a transition for which there exists a set of transitions having the same input and output places. Intuitively, the transitions removed by this rule are transitions which firing can be simulated by the firing of a sets of other transitions.

This rule is an original rule generalising the *Fusion of Parallel Transitions* rule of [Murata, 1989] and the *Abstraction of Parallel Transitions* rule of [Hichami et al., 2014]. This rule is an adaptation of the rule ϕ_S of [Desel et al., 2005] to the realm of ordinary workflow nets with no restriction on their subclasses. Indeed, outside the scope of free-choice workflow nets, additional constraints are required to ensure that the removed transitions are live. To this end, the liveness of a transition to be removed in such a way is inferred from the liveness of a source transition, a transition that, when fired, enables the transition to be removed. Note that this requirement could be relaxed. Instead of requiring the presence of a source transition, one could require the presence of a sequence of transitions, where each successive transition is enabled by the firing of the previous ones, such that the firing of this sequence of transitions enables the transition to be removed.

The inverse rule of this reduction rule is a synthesis rule able to add a single transition to an arbitrary workflow net based on its structure while preserving generalised soundness and is used to introduce choice.

The reduction rule $\phi_{RemoveT}$ is pictured in Figure 4.2 and formally described below.

Let D be a set of places, we define the function $\vartheta : D \to (P \to \mathbb{N})$ such that:

$$\forall\, d \in P,\ \vartheta(D)(d) = \begin{cases} 1, & \text{if } d \in D \\ 0, & \text{otherwise} \end{cases}$$

Let $f_1, f_2, f_3 : P \to \mathbb{N}$ be three functions, we overload the operator $+, -$ and $=$ such that $f_3 = f_1 + (-)f_2 \Leftrightarrow \forall p \in P, f_3(p) = f_1(p) + (-)f_2(p)$. Function ϑ is used to compare inputs and outputs of a set of transitions. Note here that this function does not consider self-loop transitions, a desired property as self-loop transition can be added to places (see $\phi_{RemoveST}$ reduction rule, introduced in the next subsection).

Conditions on N:	Construction of $\tilde{N}$:
– $\exists\, t \in T$	– $t^\bullet = \{op_1, .., op_{n_1}\}$
– $\exists\, G = \{g_1, .., g_n\} \subseteq T \setminus \{t\}$	– $outArc = \{t\} \times t^\bullet$
– $\vartheta_t = \vartheta(t^\bullet) - \vartheta(^\bullet t)$	– $^\bullet t = \{ip_1, .., ip_{n_2}\}$
– $\vartheta_G = \vartheta(g_1^\bullet) + .. + \vartheta(g_n^\bullet) - \vartheta(^\bullet g_1) - .. - \vartheta(^\bullet g_n)$	– $inArc = {}^\bullet t \times \{t\}$
– $\vartheta_t = \vartheta_G$	– $\tilde{P} = P$
– $\forall i, j \in \{1, .., n\}, i \neq j \Rightarrow$ $(^\bullet g_i \cap {}^\bullet g_j = g_i^\bullet \cap g_j^\bullet = \emptyset)$	– $\tilde{T} = T \setminus \{t\}$
– $(\exists\, t_s \in T \setminus (\{t\} \cup G), \forall g \in G, {}^\bullet g \subseteq t_s^\bullet) \vee (\lvert G \rvert = 1)$	– $\tilde{F} = F \setminus (inArc \cup outArc)$

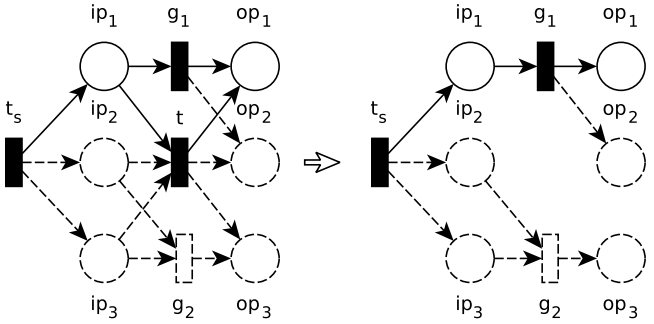


Figure 4.2: Reduction rule $\phi_{RemoveT}$ (R_2)

The soundness of $\phi_{RemoveT}$, with respect to generalised soundness, is given by the following theorem.

Theorem 22: Soundness of $\phi_{RemoveT}$

$\phi_{RemoveT}$ is a workflow net reduction rule which strongly preserves generalised soundness.

Proof. (Sketch). Let us suppose that $\vartheta_t = \vartheta_G$. By conditions on N, for all k in $\mathbb{N}$, and $\forall M^N$ in $\mathcal{R}^N(M^N_{i(k)})$, transition t is enabled if and only if transitions $g_1, .., g_n$ are also enabled. Moreover, the firing of t must result in the same marking as the successive firing of transitions $g_1, .., g_n$ in any order. It follows that $\forall k \in \mathbb{N}, M \in \mathcal{R}^N(M^N_{i(k)}), M^N_o(k) \in \mathcal{R}^N(M) \Leftrightarrow M \in \mathcal{R}^{\tilde{N}}(M^{\tilde{N}}_{i(k)}), M^{\tilde{N}}_o(k) \in \mathcal{R}^{\tilde{N}}(M)$. To conclude, suppose $|G| = 1$, then t is not dead in N if and only if g_1 is not dead in $\tilde{N}$. Alternatively, suppose $(\exists\, t_s \in T \setminus (\{t\} \cup G), \forall g \in G, {}^\bullet g \subseteq t_s^\bullet)$, then t is not dead in N if and only if t_s is not dead in $\tilde{N}$. □

R_3: Remove Self-loop

We define $\phi_{RemoveST}$, a workflow net reduction rule, which strongly preserves generalised soundness and consists in removing a transition whose input places are its output places.

This original rule generalises the *Self-Loop Transition* rule described in [Murata, 1989]. Similarly to rule $\phi_{RemoveT}$, the liveness of a transition removed by this rule also needs to be inferred from the existence of a source transition (alternatively a source sequence of transitions).

Furthermore, the inverse of this reduction rule is a synthesis rule able to add a single transition to an arbitrary workflow net based on its structure while preserving generalised soundness. It is typically used to introduce choice and repetitive tasks.

Figure 4.3 illustrates $\phi_{RemoveST}$ that is formally described below.

Conditions on N:	Construction of $\tilde{N}$:
– $\exists\, t \in T$	– $outArc = \{t\} \times t^\bullet$
– $t^\bullet = {}^\bullet t$	– $inArc = t^\bullet \times \{t\}$
– $\exists\, t_s \in T \setminus \{t\}, {}^\bullet t \subseteq t_s^\bullet \ \vee\ {}^\bullet t \subseteq {}^\bullet t_s$	– $\tilde{P} = P$
	– $\tilde{T} = T \setminus \{t\}$
	– $\tilde{F} = F \setminus (inArc \cup outArc)$

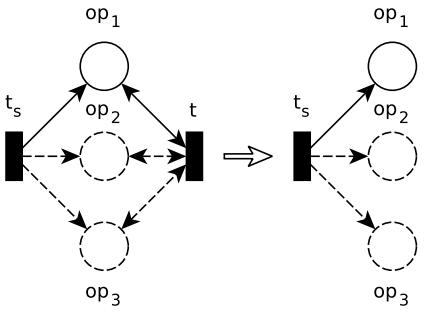


Figure 4.3: Reduction rule $\phi_{RemoveST}$ (R_3)

The soundness of $\phi_{RemoveST}$, with respect to generalised soundness, is given by the following theorem.

Theorem 23: Soundness of $\phi_{RemoveST}$

$\phi_{RemoveST}$ is a workflow net reduction rule which strongly preserves generalised soundness.

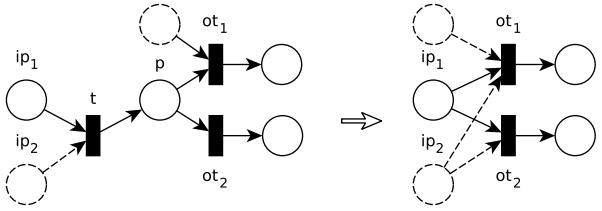
Proof. (Sketch). By conditions imposed on N, we know that the firing of transition t does not change the markings of N in which it is enabled. It follows that $\forall k \in \mathbb{N}, M \in \mathcal{R}^N(M^N_{i(k)}), M^N_o(k) \in \mathcal{R}^N(M) \Leftrightarrow M \in \mathcal{R}^{\tilde{N}}(M^{\tilde{N}}_{i(k)}), M^{\tilde{N}}_o(k) \in \mathcal{R}^{\tilde{N}}(M)$. Notice that t is not dead in N if and only if t_s is not dead in $\tilde{N}$. □

R_4: Remove Transition Place

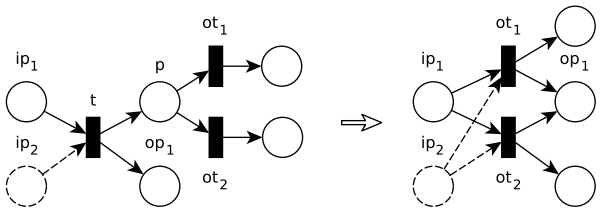
We define $\phi_{RemoveTP}$, a workflow net reduction rule, which strongly preserves generalised soundness and consists in removing a place and its only input transition. Intuitively, this rule consists in removing a place p and its only input transition t by merging transition t with the output transitions of place p.

The $\phi_{RemoveTP}$ rule is displayed in Figure 4.4(a), 4.4(b) and formally described below.

Conditions on N:	Construction of $\tilde{N}$:
– $\exists\, p \in P \setminus \{i, o\}$	– $t^\bullet \setminus p = \{op_1, .., op_{n_1}\}$
– $^\bullet p = \{t\}$	– $^\bullet t = \{ip_1, .., ip_{n_2}\}$
– $t^\bullet \neq \{p\} \Rightarrow$ $\forall\, ot \in p^\bullet, {}^\bullet ot = \{p\}$ $\wedge\, t^\bullet \cap ot^\bullet = \emptyset$	– $p^\bullet = \{ot_1, .., ot_{n_3}\}$
– $t^\bullet = \{p\} \Rightarrow$ $\forall\, ot \in p^\bullet, {}^\bullet t \cap {}^\bullet ot = \emptyset$ $\wedge\, (\exists\, ot \in p^\bullet,\ {}^\bullet ot = \{p\}$ $\vee\, \forall\, ip \in {}^\bullet t,\ ip^\bullet = \{t\})$	– $outT = \{t\} \times t^\bullet \setminus p$
	– $inT = {}^\bullet t \times \{t\}$
	– $outP = \{p\} \times p^\bullet$
	– $inArc = {}^\bullet t \times p^\bullet$
	– $outArc = p^\bullet \times t^\bullet \setminus p$
	– $\tilde{P} = P \setminus \{p\}$,
	– $\tilde{T} = T \setminus \{t\}$
	– $\tilde{F} = (F \cup inArc \cup outArc) \setminus ((t, p) \cup inT \cup outT \cup outP)$



(a) $t^\bullet = \{p\}$



(b) $t^\bullet \neq \{p\}$

Figure 4.4: Reduction rule $\phi_{RemoveTP}$ (R_4)

This rule generalises the *Post-Fusion* rule of [Berthelot, 1987, Sloan et al., 1996].

The inverse of this rule is a synthesis rule introducing a sequence of tasks (adding a task that has to be accomplished before others) and able to factor common input/output places of a set of transitions.

The soundness of $\phi_{RemoveTP}$, with respect to generalised soundness, is given by the following theorem.

Theorem 24: Soundness of $\phi_{RemoveTP}$

$\phi_{RemoveTP}$ is a workflow net reduction rule which strongly preserves generalised soundness.

Proof. (Sketch). In N the transitions $ot_1, .., ot_{n_3}$ have to consume a token in place p. All tokens consumed in place p have to be produced by transition t, which consumes a token in places $ip_1, .., ip_{n_2}$ and produces a token in places $op_1, .., op_{n_1}$ and p. Thus, $ot_1, .., ot_{n_2}$ have to consume a token in $ip_1, .., ip_{n_2}$ and produce a token in $op_1, .., op_{n_1}$. Conversely, the same analysis holds on $\tilde{N}$, we conclude that N is generalised sound if and only if $\tilde{N}$ is generalised sound. □

R_5: Remove Place Transition

We define $\phi_{RemovePT}$, a workflow net reduction rule, which strongly preserves generalised soundness and consists in removing a place and its only output transition. Intuitively this rule consist in removing a place p and its only output transition t by merging transition t with the input transitions of place p.

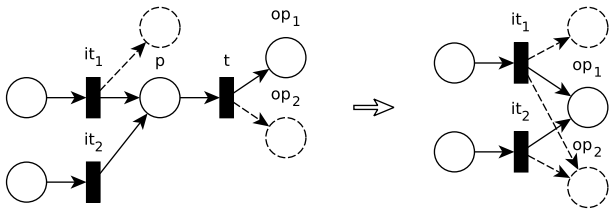
The $\phi_{RemovePT}$ rule is displayed in Figure 4.5(a), 4.5(b) and formally described below.

Conditions on N:

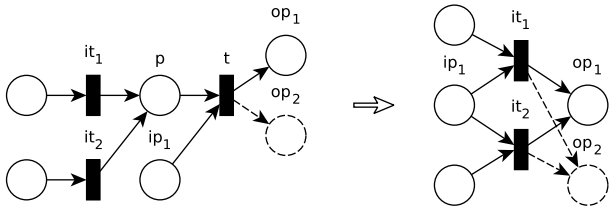
- $\exists\ p \in P \setminus \{i, o\}$
- $p^\bullet = \{t\}$
- ${}^\bullet t \neq \{p\} \Rightarrow$
  $\forall\ it \in {}^\bullet p, it^\bullet = \{p\}$
  $\wedge\ {}^\bullet t \cap {}^\bullet it = \emptyset$
  $\wedge\ ({}^\bullet it)^\bullet = \{it\}$
- ${}^\bullet t = \{p\} \Rightarrow$
  $\forall\ it \in {}^\bullet p, t^\bullet \cap it^\bullet = \emptyset$

Construction of $\tilde{N}$:

- $t^\bullet = \{op_1, .., op_{n_1}\}$
- ${}^\bullet t \setminus p = \{ip_1, .., ip_{n_2}\}$
- ${}^\bullet p = \{it_1, .., it_{n_3}\}$
- $outT = \{t\} \times t^\bullet$
- $inT = {}^\bullet t \setminus p \times \{t\}$
- $inP = {}^\bullet p \times \{p\}$
- $inArc = {}^\bullet t \setminus p \times {}^\bullet p$
- $outArc = {}^\bullet p \times t^\bullet$
- $\tilde{P} = P \setminus \{p\}$,
- $\tilde{T} = T \setminus \{t\}$
- $\tilde{F} = (F \cup inArc \cup outArc) \setminus ((p, t) \cup inT \cup outT \cup inP)$



(a) $^\bullet t = \{p\}$



(b) $^\bullet t \neq \{p\}$

Figure 4.5: Reduction rule $\phi_{RemovePT}$ (R_5)

This rule generalises the *Pre-Fusion* rule of [Berthelot, 1987, Sloan et al., 1996] as well as the reduction rule ϕ_A of [Desel et al., 2005].

The inverse of this rule is a synthesis rule introducing a sequence of tasks (adding a task that have to be accomplished after others) and able to factor common input/output places of a set of transitions.

The soundness of $\phi_{RemovePT}$, with respect to generalised soundness, is given by the following theorem.

Theorem 25: Soundness of $\phi_{RemovePT}$

$\phi_{RemovePT}$ is a workflow net reduction rule which strongly preserves generalised soundness.

Proof. (Sketch). In N the transitions $it_1, .., it_{n_3}$ have to produce a token in place p. All tokens produced in place p have to be consumed by transition t, which consumes a token in places $ip_1, .., ip_{n_2}$ and p, and produces a token in places $op_1, .., op_{n_1}$. Thus, $it_1, .., it_{n_3}$ have to consume a token in $ip_1, .., ip_{n_2}$ and produce a token in $op_1, .., op_{n_1}$. Conversely, the same analysis holds on $\tilde{N}$, we conclude that N is generalised sound if and only if $\tilde{N}$ is generalised sound. □

R_6: Remove Ring

We define $\phi_{RemoveR}$, a workflow net reduction rule, which strongly preserves generalised soundness, and consists in merging places among a ring. A ring is a set of places strongly connected by transitions with a single input place and a single output place. The transitions forming the ring are also removed. Intuitively, tokens among the places of a ring can freely move from a place of the ring to an other, therefore they might as well be on the same place.

This rule is an original one, its inverse rule is a synthesis rule which transforms a place into a ring, distributing its input and output transitions among the places of the introduced ring.

Figure 4.6 illustrates $\phi_{RemoveR}$ that is formally described below.

Conditions on N:

- $\exists\, \{p_1, .., p_n\} \subseteq P$
- $\exists\, \{t_1, .., t_m\} \subseteq T$
- $\forall\, i \in \{1, .., m\}, |{}^\bullet t_i| = |t_i^\bullet| = 1$
- $\forall\, i, j \in \{1, .., n\}, {}^\bullet p_i \cap {}^\bullet p_j = p_i^\bullet \cap p_j^\bullet = \emptyset$
- $\forall\ i, j \in \{1, .., m\}$, $\exists\ \sigma : \{1, .., k\} \rightarrow \{p_1, .., p_n\} \cup \{t_1, .., t_m\}$ a path of length k such that $\sigma(1) = p_i \wedge \sigma(k) = p_j \wedge \forall\ x \in \{1, .., k-1\}, (\sigma(x), \sigma(x+1)) \in F$

Construction of $\tilde{N}$:

- $ringArc = (\{p_1, .., p_n\} \times \{t_1, .., t_m\}) \cup (\{t_1, .., t_m\} \times \{p_1, .., p_n\})) \cap F$
- $inT = {}^\bullet p_1 \cup .. \cup {}^\bullet p_n$
- $outT = p_1^\bullet \cup .. \cup p_n^\bullet$
- $removedA = ((inT \times \{p_1, .., p_n\}) \cup (\{p_1, .., p_n\} \times outT)) \cap F$
- $addA = (inT \times p) \cup (p \times outT)$
- $\tilde{P} = (P \cup p) \setminus \{p_1, .., p_n\}$
- $\tilde{T} = T \setminus \{t_1, .., t_m\}$
- $\tilde{F} = (F \cup addA) \setminus removedA$

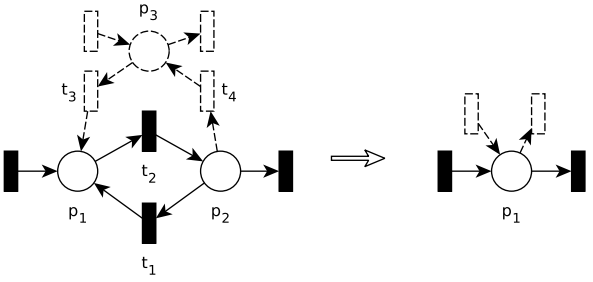


Figure 4.6: Reduction rule $\phi_{RemoveR}$ (R_6)

The soundness of $\phi_{RemoveR}$, with respect to generalised soundness, is given by the following theorem.

Theorem 26: Soundness of $\phi_{RemoveR}$

$\phi_{RemoveR}$ is a workflow net reduction rule which strongly preserves generalised soundness.

Proof. (Sketch). By conditions imposed on N, tokens among the places $p_1, .., p_n$ of a ring can freely move from a place of the ring to an other by firing a sequence of transitions formed with transitions $t_1, .., t_m$. It follows that each token produced (resp. consumed) by an input (resp. output) transition of a place in the ring will eventually be (resp. has been), after (resp. before) the firing of a possibly empty sequence of transitions formed with transitions $t_1, .., t_m$, consumed (resp. produced) by any output (resp. input) transitions of a place of the ring. Likewise, in $\tilde{N}$ each token produced (resp. consumed) by an input (resp. output) transition of p will be (resp. has been) consumed (resp. produced) by an output (resp. input) transition of p. It follows that N is generalised sound if and only if $\tilde{N}$ is generalised sound. □

In this section we defined six reduction rules which together constitute a generic generalised soundness preserving reduction kit. These rules generalise the rules previously presented in the literature [Berthelot, 1987, Murata, 1989, Sloan et al., 1996, Voorhoeve et al., 1997, Sadiq et al., 2000, Lin et al., 2002, Desel et al., 2005, Hichami et al., 2014] thereby extending the range of workflow nets reducible in such a way.

The next sections present two applications of this reduction kit: a generalised semi-decision procedure and a pre-processing step towards the verification of (extended) modal specifications.

4.2/ Semi-Decision of Generalised Soundness

This section proposes an algorithm based on the workflow nets reduction *kit* Φ^* described in the previous section, for semi-deciding the generalised soundness of workflow nets.

Our approach is based on Lemma 4 page 36. This lemma allows us to infer the validity of a property of a workflow net from its transformed instance as long as the transformation rules applied to obtain it strongly preserve this property.

The reduction *kit* Φ^* strongly preserves generalised soundness. Let N and $\tilde{N}$ be two workflow nets such that $(N, \tilde{N}) \in \Phi^*$ then the workflow N is generalised sound if and only if the workflow net $\tilde{N}$ is generalised sound.

Furthermore, it is trivial that the workflow net $N_{Atomic} = \langle \{i, o\}, \{t\}, \{(i, t), (t, o)\} \rangle$ (i.e. a workflow net composed of a single transition whose input place is the initial place and output place is the final place) is generalised sound.

It follows that if $(N, N_{Atomic}) \in \Phi^*$ then the workflow net N is generalised sound. Unfortunately, the reduction *kit* Φ^* is not complete with respect to generalized soundness over ordinary workflow nets. This leads to the design of an algorithm to semi-decide whether a workflow net N is generalised sound.

This algorithm proceeds by iteratively trying to apply any of the reduction rules of Φ^* to the input the workflow net N until a fix-point is reached (none of the reduction rules can be applied). If the workflow net produced by these reductions equals N_{Atomic}, one can conclude that N is generalised sound. Otherwise, one cannot directly conclude about generalised soundness, but the reduced workflow net is saved to be further analysed using classical techniques such as model-checking.

This procedure is described by Algorithm 2 which is based on: (i) the set of workflow net reduction rules $\Phi = \{R_1, .., R_6\}$, (ii) an auxiliary function $size(N)$, which returns the number of nodes of a workflow net N at each iteration step, (iii) a function $TryApplyRule(\phi, N)$, which returns either $\tilde{N}$ if the rule ϕ can be applied to N to produce $\tilde{N}$, or N otherwise, and (iv) $save(N)$, a function that saves N.

Theorem 27: Termination of Algorithm 2

The procedure described by Algorithm 2 terminates.

Proof. (Sketch). Every rule applied by Algorithm 2 strictly reduces the number of nodes of N. None of the applied rules can produce a workflow net with less than one node. Thus it always terminates when no workflow net reduction rules can be applied, and either provides an atomic sound net or saves the reduced workflow net. □

```
Data: N = ⟨P, T, F⟩
Result: Generalised soundness of N
int sizeN = 0;
do
    sizeN = size(N);
    forall the φ ∈ Φ do
        int subsizeN = 0;
        do
            subsizeN = size(N);
            N = TryApplyRule(φ, N);
        while size(N) < subsizeN;
    end
while size(N) < sizeN;
if N = N_Atomic then
    return true;
else
    save(N);
    return unknown;
end
```

Algorithm 2: Generalised soundness semi-decision algorithm

Theorem 28: Soundness of Algorithm 2

The procedure described by Algorithm 2 is sound.

Proof. (Sketch). The set of workflow net reduction rules strongly preserves generalised soundness. By Theorem 4 page 36, the procedure described by Algorithm 2 is sound. □

The application of this algorithm is illustrated by Figure 4.7. This figure depicts the sequence of workflow nets obtained by applying the reduction rules indicated by the labels of the large arrows linking them. The large arrows labelled R_i^* indicate that the transformation R_i is sequentially applied many times to the source workflow net to produce the target one. The nodes coloured in red are the nodes of the source workflow net that are deleted by the reduction rule(s) applied to produce the target one. Whenever the applied rule is R_1 (resp. R_2) the places (resp. transitions) which are equivalent to the deleted node(s) are coloured in green. The last workflow net obtained is the atomic workflow net. Therefore, we can conclude that each workflow net depicted by Figure 4.7 is generalised sound.

We described how the workflow nets reduction *kit* described in Section 4.1 page 68 can be used to semi-decide the generalised soundness of workflow nets. Moreover, it is important to note that, in the case where this procedure cannot conclude, all was not done in vein as the produced workflow net size is most likely reduced. Furthermore, useful error diagnostics are given in the form of an irreducible graph. This makes the presented procedure a valuable pre-processing step towards further workflow net soundness analysis.

The following section reuses this concept and defines restrictions on the workflow nets reduction *kit* described in Section 4.1 page 68. These restrictions enable them to preserve the (in)validity of (extended) modal specifications in order to be used as a pre-processing set towards their verification.

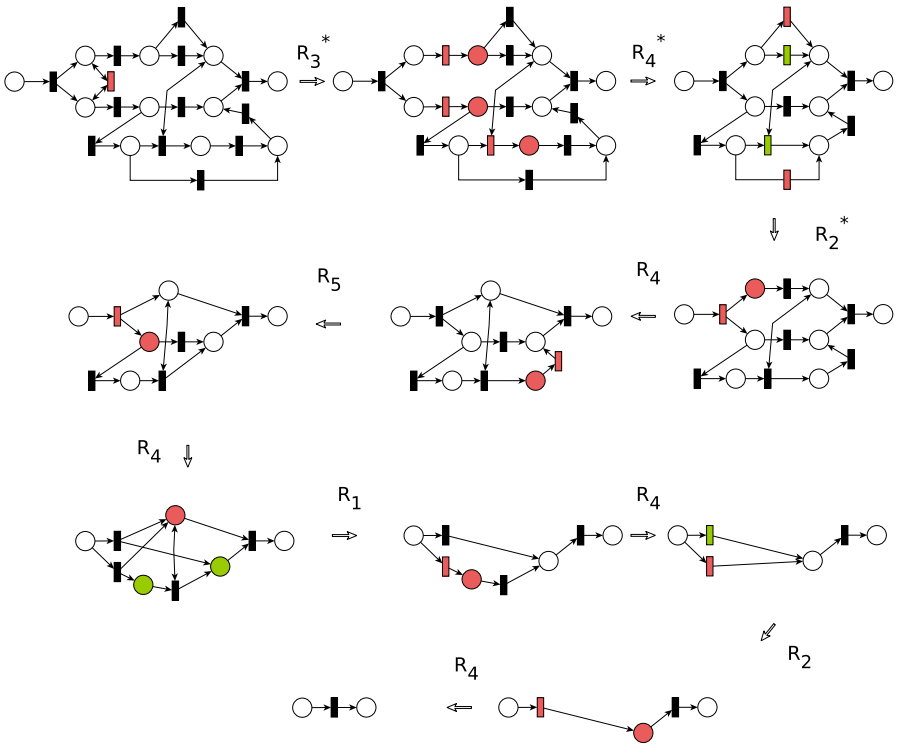


Figure 4.7: Illustration of the application of Algorithm 2

4.3/ Preprocessing Modal Specification Verification

In Section 3.1.3 page 53 we presented a verification method to verify the (in)validity of extended modal specifications (Section 3.1.1 page 44) through the evaluation of the satisfiability of constraint systems. This method principally relies on the definition of over-approximations (e.g., the state equation) which are then refined through the composition of under-approximations. However, the used over-approximations are quite complex. For example, the existence of a solution to the state equation (Section 2.2.2.1 page 30) of a workflow net is an NP-complete problem. This has the effect of making this method intractable for large workflow nets.

Nonetheless, in Section 2.1.9 page 25, a vertical abstraction mechanism is defined through two vertical abstraction operations: *Place refinement* and *Transition refinement*. This abstraction mechanism leads to the definition of hierarchical workflow nets representing workflow nets with multiple layers of detail. When the analysed workflow net is developed by stepwise refinement using *Place refinement* and *Transition refinement*, this hierarchical workflow net is explicitly given by the modellers. In a broader context, hierarchical workflow nets can be automatically and efficiently constructed (e.g., by detecting strongly connected components) from workflow nets through their decomposition [Polyvyanyy et al., 2011, Vanhatalo et al., 2007, Koehler et al., 2014].

The presented extended modal specification verification method, as most analysis tools, is not able to directly deal with hierarchical workflow nets. Instead, it takes as input the underlying workflow nets of hierarchical workflow nets. However, not all levels of detail of a hierarchical workflow net are required in order to infer the (in)validity of a given extended modal specification. It follows that the construction of a dedicated underlying workflow net, which preserves the (in)validity of a given extended modal specification, constitutes a vertical abstraction mechanism able to reduce the size of the workflow net under analysis.

Furthermore, in Section 4.1 page 68, a reduction *kit* able to abstract workflow nets in order to reduce their size while preserving generalised soundness is presented. It allows an horizontal abstraction of workflow nets at an higher level. This reduction *kit* is particularly well-adapted to the reduction of workflow nets designed through refinement, a development approach which notably uses modal specifications.

It follows that adapting the reductions rules used by this reduction *kit* in order to preserve the (in)validity of a given extended modal specification constitutes an horizontal abstraction mechanism able to reduce the size of the workflow net under analysis.

Such abstraction mechanisms, used as pre-processing steps of the verification of a given extended modal specification, enlarge the range of workflow nets that can be analysed. Indeed, due to the exponential nature of the verification method with respect to the size of the workflow net under analysis, any reduction of the size of the workflow net under analysis should significantly improve the efficiency of the verification method.

Conceptually, this kind of approaches can be seen as a slicing approach [Rabbi et al., 2013] since they aim at performing abstraction guided by a specific specification formula (equivalently a set of specification formulae). Indeed, slicing is a technique to syntactically reduce a model in such a way that at best the reduced model contains only those parts that may influence the property the model is analysed for [Rakow, 2008], which is the intent of the proposed reduction methods.

The remaining of this section is divided into two sub-sections presenting reduction methods preserving the (in)validity of a given extended modal specification. The first sub-section presents a reduction procedure based on the hierarchical representation of a workflow net. The second one details a reduction method based on the reduction rules presented in Section 4.1 page 68.

4.3.1/ Reduction based on hierarchical workflow nets

This section presents a reduction method preserving the (in)validity of a given extended modal specification based on the hierarchical representation of workflow nets.

We begin by defining *subTransitions*, an utility function described in Algorithm 3, which takes as input a hierarchical workflow net node (i.e. a hierarchical workflow net) and returns the set of transitions of its underlying workflow net.

Data: $W = \langle P_h, T_h, F_h \rangle$ a hierarchical workflow net
Result: T a set of transitions
Function subTransitions($W = \langle P_h, T_h, F_h \rangle$)
  $T = T_h$;
  forall the $S = \langle P_s, T_s, F_s \rangle \in P_h \cup T_h$ **do**
    if $T_s \neq \emptyset$ **then**
      $T = T \cup T_s \cup subTransitions(S)$;
    end
  end
  return T

Algorithm 3: Definition of the utility function *subTransitions*

We now define $underlying_r$, a function described in Algorithm 4. This function takes as inputs a hierarchical workflow net and an extended modal specification formula, and returns the underlying workflow net of the input hierarchical workflow net restricted to the layers of detail relevant to the input extended modal specification formula. Differently from the function *underlying*, described

in Algorithm 1 page 27, the function $underlying_r$ does not recursively expand every node of the input hierarchical workflow net, instead it only chooses to expand nodes for which the set produced by $subTransitions$ contains a transition in the domain of the input extended modal specification formula.

Data: $W = \langle P_h, T_h, F_h \rangle$ a hierarchical workflow net
m an extended modal specification formula
Result: $N = \langle P, T, F \rangle$ the workflow net underlying W restricted to the layers of detail relevant to n
Function $underlying_r(W = \langle P_h, T_h, F_h \rangle, m)$
  $N = \langle P_h, T_h, F_h \rangle$;
  forall the $S = \langle P_s, T_s, F_s \rangle \in P_h \cup T_h$ **do**
    if $T_s \neq \emptyset \wedge (subTransitions(S) \cap Domain(m)) \neq \emptyset$ **then**
      $N = N \otimes_S underlying_r(S)$;
    end
  end
  return N

Algorithm 4: Definition of the function $underlying_r$

Let $W = \langle P_h, T_h, F_h \rangle$ be a hierarchical workflow net, and m a given extended modal specification formula.

The extended modal specification formula m, interpreted as either a *may*-formula or a *must*-formula, is said to be (in)valid over the hierarchical workflow net W if and only if it is (in)valid over the $underlying(W)$ workflow net.

The reduction procedure based on the function $underlying_r$ is said to be sound with respect to (in)validity of m if and only if $underlying(W) \models_{must} m \Leftrightarrow underlying_r(W, m) \models_{must} m$, and $underlying(W) \models_{may} m \Leftrightarrow underlying_r(W, m) \models_{may} m$.

In order for the reduction procedure based on the function $underlying_r$ to be sound with respect to (in)validity of m, the following assumptions are required:

$$\forall\, t \in Domain(m),\ subTransitions(t) \cap Domain(m) = \emptyset, \text{ and}$$

$$W \text{is generalised sound}$$

The first assumption concerns the given modal specification formula m: it ensures that transitions in $Domain(m)$ are not deleted by refinement during the construction of the underlying workflow net. The second assumption concerns the hierarchical workflow net W: it guarantees that the abstracted layers of detail always produce valid (sub)executions.

Theorem 29: Soundness of $underlying_r$

Let $W = \langle P_h, T_h, F_h \rangle$ be a generalised sound hierarchical workflow net, and m a given extended modal specification formula such that $\forall t \in Domain(m), subTransitions(t) \cap Domain(m) = \emptyset$.
The reduction procedure based on the function $underlying_r$ is sound with respect to (in)validity of m: $underlying(W) \models_{must} m \Leftrightarrow underlying_r(W, m) \models_{must} m$, and $underlying(W) \models_{may} m \Leftrightarrow underlying_r(W, m) \models_{may} m$.

Proof. (Sketch). We proceed by showing that the workflow nets $underlying(W) = \langle P_u, T_u, F_u \rangle$ and $underlying_r(W, m) = \langle P_r, T_r, F_r \rangle$ are weak trace equivalent with respect to the transitions set $Domain(m)$.

Note that, assuming W does not refine the transition of $Domain(m)$, $Domain(m) \subseteq T_u$ and $Domain(m) \subseteq T_r$ by definitions.

Further, transition $t \in T_r \setminus T_u$ is an abstract transition which always corresponds to correct executions of $underlying(t)$ as W is generalised sound. Likewise, place $p \in P_r \setminus P_u$ is an abstract place and transition $t \in p^\bullet$ corresponds to correct executions composed of correct executions of $underlying(p)$ followed by t.

It follows that, $underlying(W)$ and $underlying_r(W, m)$ are weak trace equivalent with respect to the transitions set $Domain(m)$. We can conclude that $underlying(W) \models_{must} m \Leftrightarrow underlying_r(W, m) \models_{must} m$, and $underlying(W) \models_{may} m \Leftrightarrow underlying_r(W, m) \models_{may} m$. □

Theorem 29 allows us to conclude that, in order to verify a modal specification formula m interpreted as either a *may*-formula or a *must*-formula over a generalised sound hierarchical workflow net W, it is sufficient to consider the workflow net $underlying_r(W, m)$ rather than the workflow net $underlying(W)$. As the workflow net $underlying_r(W, m)$ can be considerably smaller than the workflow net $underlying(W)$, this reduction approach constitutes a valuable pre-processing step towards the verification of m interpreted ether as a *may*-formula or a *must*-formula.

4.3.2/ Reduction based on reduction rules

This section presents a reduction method preserving the (in)validity of a given extended modal specification based on the reduction rules presented in Section 4.1 page 68.

More precisely, it presents restrictions imposed on the reduction rules presented in Section 4.1 page 68. Given an extended modal specification formula m, these restricted reduction rules preserve the (in)validity of the extended modal specification formula m interpreted as either a *may*-formula or a *must*-formula.

In what follows, the six workflow net reduction rules presented in Section 4.1 page 68 are reviewed. The restrictions added to the *conditions of application*, under which it can be applied to a source workflow net $N = \langle P, T, F \rangle$ to produce a target workflow net $\tilde{N} = \langle \tilde{P}, \tilde{T}, \tilde{F} \rangle$, are specified with regard to a given extended modal specification formula m.

R_1: Remove Place

We define $\phi_{RemoveP}(m)$ a reduction rule strictly equivalent to the reduction $\phi_{RemoveP}$ defined page 69. No further restriction to *conditions of application* of this rule is required in order to enable it to strongly preserve the (in)validity of the extended modal specification formula m. Indeed, places removed by this reduction rule are redundant places that do not change the executions set of a workflow net.

Theorem 30: Soundness of $\phi_{RemoveP}(m)$

$\phi_{RemoveP}(m)$ is a workflow net reduction rule which strongly preserves the (in)validity of the extended modal specification formula m interpreted as either a *may*-formula or a *must*-formula.

Proof. (Sketch). We proceed by showing that the workflow nets N and $\tilde{N}$ are trace equivalent.

Let $f : (\tilde{P} \to \mathbb{N}) \to (P \to \mathbb{N})$ be a bijective function such that $f(M)(g) = M(g)$ for all $g \in \tilde{P}$ and $f(M)(p) = M(g_1) + .. + M(g_n)$.

Let $l \in \mathbb{N}$ and $\sigma = t_1, .., t_l$ be a transition sequence of size l.

By conditions on N, every transition that produces (resp. consumes) a token in any of the places of G also produces (resp. consumes) a token in p.

It follows that there exists a sequence $M_1, .., M_{l+1}$ of $l + 1$ markings of $\tilde{N}$ such that $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} .. \xrightarrow{t_l} M_{l+1}$ is an execution of $\tilde{N}$ if and only if $f(M_1) \xrightarrow{t_1} f(M_2) \xrightarrow{t_2} .. \xrightarrow{t_l} f(M_{l+1})$ is an execution of N.

Therefore, the workflow nets N and $\tilde{N}$ are trace equivalent. We can then conclude that $N \models_{must} m \Leftrightarrow \tilde{N} \models_{must} m$, and $N \models_{may} m \Leftrightarrow \tilde{N} \models_{may} m$. □

R_2: Remove Transition

We define $\phi_{RemoveT}(m)$ a reduction rule which further constrains the *conditions of application* of the $\phi_{RemoveT}$ reduction rule defined page 70.

In order to ensure that $\phi_{RemoveT}(m)$ strongly preserves the (in)validity of the extended modal specification formula m, interpreted as either a *may*-formula or a *must*-formula, one has to make sure that the removed transition is not part of the domain of m (i.e. $t \notin Domain(m)$). Further, one also has to ensure that $\{g_1, .., g_n\} \cap Domain(m) = \emptyset$ to guarantee that alternative behaviours including transition of the domain of m are not lost by the application of such a transformation rule.

To this end, the following constraint is added to the *conditions of application* of the $\phi_{RemoveT}$ reduction rule:

$$t \notin Domain(m) \wedge \{g_1, .., g_n\} \cap Domain(m) = \emptyset$$

Theorem 31: Soundness of $\phi_{RemoveT}(m)$

$\phi_{RemoveT}(m)$ is a workflow net reduction rule which strongly preserves the (in)validity of the extended modal specification formula m interpreted as either a *may*-formula or a *must*-formula.

Proof. (Sketch). We proceed to show that the workflow nets N and $\tilde{N}$ are weak trace equivalent with respect to the transitions set $Domain(m)$. By condition imposed on N, the firing of transition t is equivalent to the successive firings of the transitions sequence $g_1, .., g_n$. It follows that the occurrences of t in the executions of N can be replaced by the transitions sequence $g_1, .., g_n$ to form executions of $\tilde{N}$. Further, any execution of $\tilde{N}$ is an execution of N. As $t \notin Domain(m)$ and $\{g_1, .., g_n\} \cap Domain(m) = \emptyset$, we can conclude that N and $\tilde{N}$ are weak trace equivalent with respect to the transitions set $Domain(m)$. It follows that $N \models_{must} m \Leftrightarrow \tilde{N} \models_{must} m$, and $N \models_{may} m \Leftrightarrow \tilde{N} \models_{may} m$. □

R_3: Remove Self-loop

We define $\phi_{RemoveST}(m)$ a reduction rule which further constrains the *conditions of application* of the $\phi_{RemoveST}$ reduction rule defined page 72.

In order to ensure that $\phi_{RemoveST}(m)$ strongly preserves the (in)validity of the extended modal specification formula m, interpreted as either a *may*-formula or a *must*-formula, one has to make sure that the removed transition is not part of the domain of m (i.e. $t \notin Domain(m)$).

To this end, the following constraint is added to the *conditions of application* of the $\phi_{RemoveST}$ reduction rule:

$$t \notin Domain(m)$$

Theorem 32: Soundness of $\phi_{RemoveST}(m)$

$\phi_{RemoveST}(m)$ is a workflow net reduction rule which strongly preserves the (in)validity of the extended modal specification formula m interpreted as either a *may*-formula or a *must*-formula.

Proof. (Sketch). We proceed by showing that the workflow nets N and $\tilde{N}$ are weak trace equivalent with respect to the transitions set $Domain(m)$.

By condition imposed on N, firing transition t does not change the marking in which it is fired. Therefore any execution of $\tilde{N}$ is also an execution of N. Further, any execution of N where every occurrence of t is removed is an execution of $\tilde{N}$. As $t \notin Domain(m)$ we can conclude that N and $\tilde{N}$ are weak trace equivalent with respect to the transitions set $Domain(m)$.

It follows that $N \models_{must} m \Leftrightarrow \tilde{N} \models_{must} m$, and $N \models_{may} m \Leftrightarrow \tilde{N} \models_{may} m$. □

R_4: Remove Transition Place

We define $\phi_{RemoveTP}(m)$ a reduction rule which further constrains the *conditions of application* of the $\phi_{RemoveTP}$ reduction rule defined page 73.

In order to ensure that $\phi_{RemoveTP}(m)$ strongly preserves the (in)validity of the extended modal specification formula m, interpreted as either a *may*-formula or a *must*-formula, one has to ensure that the removed transition is not part of the domain of m (i.e. $t \notin Domain(m)$).

To this end, the following constraint is added to the *conditions of application* of the $\phi_{RemoveTP}$ reduction rule:

$$t \notin Domain(m)$$

Theorem 33: Soundness of $\phi_{RemoveTP}(m)$

$\phi_{RemoveTP}(m)$ is a workflow net reduction rule which strongly preserves the (in)validity of the extended modal specification formula m interpreted as either a *may*-formula or a *must*-formula.

Proof. (Sketch). As before, we show that the workflow nets N and $\tilde{N}$ are weak trace equivalent with respect to the transitions set $Domain(m)$.

By construction of $\tilde{N}$, $\forall\ i \in \{1, .., n_3\}$, the firing of ot_i in $\tilde{N}$ is equivalent to the firing of t, ot_i in N.

Therefore any execution of $\tilde{N}$ where, for every $i \in \{1, .., n_3\}$, every occurrence of ot_i is replaced by the transitions sequence t, ot_i, is an execution of N. Further, any execution of N where every occurrence of t is removed is an execution of $\tilde{N}$. As $t \notin Domain(m)$, we can conclude that N and $\tilde{N}$ are weak trace equivalent with respect to the transitions set $Domain(m)$.

It follows that $N \models_{must} m \Leftrightarrow \tilde{N} \models_{must} m$, and $N \models_{may} m \Leftrightarrow \tilde{N} \models_{may} m$. □

R_5: Remove Place Transition

We define $\phi_{RemovePT}$ a reduction rule which further constrains the *conditions of application* of the $\phi_{RemovePT}$ reduction rule defined page 74.

Similarly to the constraint added to $\phi_{RemoveTP}$ to define $\phi_{RemoveTP}(m)$, to ensure that $\phi_{RemovePT}(m)$ strongly preserves the (in)validity of the extended modal specification formula m, interpreted as either a *may*-formula or a *must*-formula, one has to make sure that the removed transition is not part of the domain of m (i.e. $t \notin Domain(m)$).

To this end, the following constraint is added to the *conditions of application* of the $\phi_{RemovePT}$ reduction rule:

$$t \notin Domain(m)$$

Theorem 34: Soundness of $\phi_{RemovePT}(m)$

$\phi_{RemovePT}(m)$ is a workflow net reduction rule which strongly preserves the (in)validity of the extended modal specification formula m interpreted as either a *may*-formula or a *must*-formula.

Proof. (Sketch). Similarly to previous proofs, we show that the workflow nets N and $\tilde{N}$ are weak trace equivalent with respect to the transitions set $Domain(m)$.

By construction of $\tilde{N}$, $\forall\ i \in \{1, .., n_3\}$, the firing of it_i in $\tilde{N}$ is equivalent to the firing of it_i, t in N.

Therefore any execution of $\tilde{N}$ where, for every $i \in \{1, .., n_3\}$, every occurrence of it_i is replaced by the transitions sequence it_i, t, is an execution of N. Further, any execution of N where every occurrence of t is removed is an execution of $\tilde{N}$. As $t \notin Domain(m)$, we can conclude that N and $\tilde{N}$ are weak trace equivalent with respect to the transitions set $Domain(m)$.

It follows that $N \models_{must} m \Leftrightarrow \tilde{N} \models_{must} m$, and $N \models_{may} m \Leftrightarrow \tilde{N} \models_{may} m$. □

R_6: Remove Ring

It remains to define $\phi_{RemoveR}(m)$, a reduction rule which further constrains the *conditions of application* of the $\phi_{RemoveR}$ reduction rule defined page 75.

In order to ensure that $\phi_{RemoveR}(m)$ strongly preserves the (in)validity of the extended modal specification formula m, interpreted as either a *may*-formula or a *must*-formula, one has to make sure that none of the removed transitions (i.e. the transitions of the ring) is part of the domain of m (i.e. $\{t_1, .., t_m\} \cap Domain(m) = \emptyset$).

To this end, the following constraint is added to the *conditions of application* of the $\phi_{RemovePT}$ reduction rule:

$$\{t_1, .., t_m\} \cap Domain(m) = \emptyset$$

Theorem 35: Soundness of $\phi_{RemoveR}(m)$

$\phi_{RemoveR}(m)$ is a workflow net reduction rule which strongly preserves the (in)validity of the extended modal specification formula m interpreted as either a *may*-formula or a *must*-formula.

Proof. (Sketch). As previously, we proceed by showing that the workflow nets N and $\tilde{N}$ are weak trace equivalent with respect to the transitions set $Domain(m)$.

By condition imposed on N, every execution of N where occurrences of the transitions $t_1, .., t_m$ are removed is also an execution of $\tilde{N}$. Further, every execution of $\tilde{N}$ where occurrences of a transition $t \in outT$ are replaced by the transition sequence seq, t, where seq is a sequence of transitions involving transitions of the ring such that it enables t, is an execution of N. As $\{t_1, .., t_m\} \cap Domain(m) = \emptyset$, we can conclude that N and $\tilde{N}$ are weak trace equivalent with respect to the transitions set $Domain(m)$.

It follows that $N \models_{must} m \Leftrightarrow \tilde{N} \models_{must} m$, and $N \models_{may} m \Leftrightarrow \tilde{N} \models_{may} m$. □

For any given extended modal specification formula m, the six presented rules thus defines together a reduction *kit*, called $\Phi^*(m)$, which strongly preserves the (in)validity of the extended modal specification formula m interpreted as either a *may*-formula or a *must*-formula. Indeed, given N and $\tilde{N}$ two workflow nets such that $(N, \tilde{N}) \in \Phi^*(m)$, we know that $N \models_{must} m \Leftrightarrow \tilde{N} \models_{must} m$, and $N \models_{may} m \Leftrightarrow \tilde{N} \models_{may} m$.

We can conclude that, given an extended modal specification formula m, the reduction *kit* $\Phi^*(m)$ defined in this section can be used as a pre-processing step toward the verification of the extended modal specification formula m interpreted as either a *may*-formula or a *must*-formula. Indeed, beside reducing the size of the workflow net to be analysed, this pre-processing step provides useful diagnostic information in the form of an irreducible graph.

Furthermore, note that the reduction rules of $\Phi^*(m)$ preserve weak trace equivalence with respect to the transitions set $Domain(m)$. They are therefore also candidate to a pre-processing step toward the verification of other behavioural properties (e.g., LTL on traces [Westergaard, 2011], CTL on traces) expressed over $Domain(m)$.

4.4/ Synthesis

This chapter presented reduction methods – abstraction methods – that have the ability to reduce the size of workflow nets while strongly preserving properties of interest.

It first presented six reduction rules generalising existing reduction rules defined by previous work [Berthelot, 1987, Murata, 1989, Sloan et al., 1996, Voorhoeve et al., 1997, Sadiq et al., 2000, Lin et al., 2002, Desel et al., 2005, Hichami et al., 2014]. It is proven that these reduction rules strongly preserve generalised soundness over workflows nets, an essential and necessary correctness property that must be satisfied by workflow nets.

Based on the definition of these reduction rules, a generalised soundness semi-decision is then described. It notably showed that, despite not being complete, this procedure, whenever unable to

conclude, can be used as a pre-processing step toward verification of generalised soundness. Indeed, by reducing the size of the analysed workflow nets, this procedure can enhance conventional generalised soundness verification methods.

Finally, two reduction methods preserving the (in)validity of a given extended modal specification formula interpreted as either a *may*-formula or a *must*-formula are described. The first of these method – based on the hierarchical representation of workflow nets – aims at abstracting unnecessary layers of detail. The second method is based on the six reduction rules previously presented. Both reduction methods were proved to be sound. Thanks to the potential size reduction they provide, these reduction methods constitute pre-processing steps for extended modal specifications verification approaches (e.g., the one presented in Section 3.1.3 page 53).

Now that the theoretical contributions of this thesis have been described, the following chapter presents the tools that have been implemented as well as experimentations that have been carried out over industrial workflow nets in order to validate the approaches introduced in this chapter and the previous one.

5

Experimental Evaluation

"The proper method for inquiring after the properties of things is to deduce them from experiments."

— Isaac Newton

Contents

5.1 Study Cases 90

5.1.1 Issue Tracking System 90

5.1.2 Question and Answer Portal 91

5.1.3 Tax Accounting Manager 95

5.2 Tool Chain Implementation 98

5.2.1 Modal Specification Verifier 98

5.2.2 Reduction Tool 103

5.3 Study cases results 104

5.4 Scalability 108

5.4.1 Benchmark's Generation Tool 108

5.4.2 Experimental Evaluation of Modal Specification Verification 109

5.4.3 Experimental Evaluation of Reduction Methods 117

5.5 Synthesis 123

This chapter describes experimental results obtained in order to assess the value of the contributions presented previously as well as the tools developed to support the proposed approach. The first section introduces study cases (i.e. real-life examples) which highlight and motivate the use of workflow nets and extended modal specifications. The second section describes the dedicated tool chain implemented to carry out the verification of extended modal specifications according to the approach presented in Section 3.1.3 page 53 as well as the reduction procedures of workflow nets presented in Sections 4.2 page 77 and 4.3 page 79. The third section exposes and discusses experimental results obtained over the study cases presented in the first section. The fourth section has for objective to demonstrate the scalability of the proposed approaches over workflow nets of growing size and complexity. It first describes a workflow nets benchmarks generation tool able to produce large benchmarks of workflow nets of growing size and complexity together with their (in)valid extended modal specifications. It then proceeds to expose and discuss experimental results obtained over such benchmarks before providing an experimental evaluation of the effectiveness and scalability of the reduction methods proposed in Sections 4.2 page 77 and 4.3 page 79. Finally, as a conclusion, major results are summarised.

5.1/ Study Cases

This section presents the three study cases considered during this thesis to validate and evaluate the extended modal specification verification approach described in Chapter 3 page 43. These three study cases are real-life examples of industrial workflows obtained through collaboration with industrial actors.

5.1.1/ Issue Tracking System

The first study case concerns a proprietary issue tracking system used to manage bugs and issues requested by the customers of a tool provider company[1].

General Description

This issue tracking system enables the provider to create, update and drop tickets reporting on customer's issues, and thus provides knowledge base containing problem definition, information about customer's environment, improvements and solutions to common problems, request status, request priority, and other relevant data needed to efficiently manage all the company projects. It must also be compliant with respect to several rules ensuring that business processes are suitable as well as streamlined, and implement best practices to increase management effectiveness.

Modelling

Figure 5.1 depicts an excerpt of the corresponding business process—specified from textual requirements by a business analyst team of the company—modelled using a Petri net workflow. Note that this workflow was first designed using the *BPMN Standard* [White, 2004] and has been transformed into a hierarchical workflow net following [Raedts et al., 2007]'s approach.

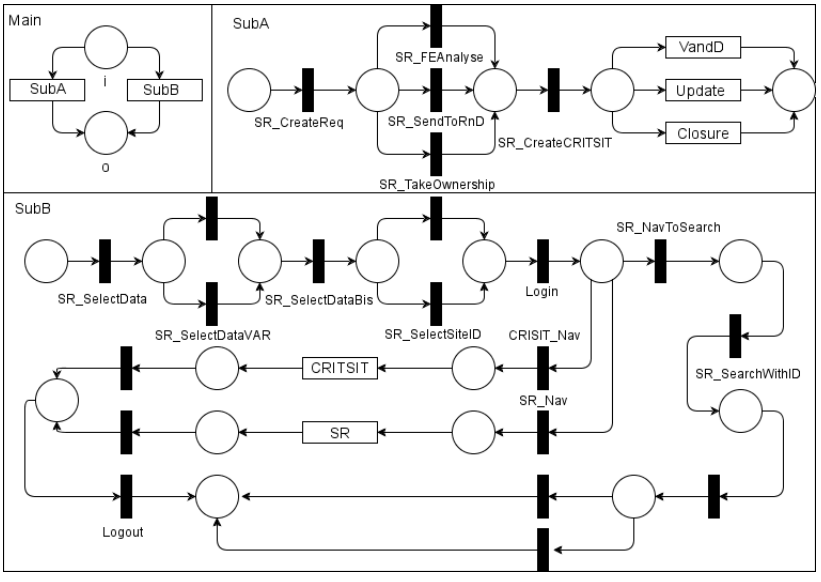


Figure 5.1: Excerpt of issue tracking system workflow net

[1]For confidentiality reasons, some details about this case-study are not given.

The main process, in the top left model, is defined by two possible distinct scenarii (*SubA* and *SubB*), which are described by two other workflow nets. In the figure, big rectangles (as for *SubA* and *SubB*) define other workflow nets. Some of them are not presented here: this example is indeed deliberately simplified and abstracted to allow its small and easily understandable presentation; its complete and fully detailed workflow net is composed of 165 places and 204 transitions.

Requirements

This industrial business workflow is directly driven by the need to verify some behavioural properties possibly at the early stage of development life cycle, before going to implementation.

For this business process, the goal is to verify, at the specification or design stage of the development, some required behavioural properties, derived from textual requirements and business analyst expertise. Such properties are presented in Table 5.1 and denoted p_i for later references.

#	Property
p_1	During a session, either the scenario *SubA* or the scenario *SubB* (and not both of them) must be executed
p_2	When the scenario *SubB* is considered then the user must login
p_3	Once a critical situation request is pending, it can either be updated, validated and dispatched, or closed
p_4	Once a critical situation is created, it can be updated and closed
p_5	At any time, a service request can be upgraded to a critical situation request
p_6	A logged user must logout to exit the current session

Table 5.1: *Issue tracking system* requirements

To ensure the specified business process model verifies this kind of business rules, there is a need to express and assess them using modal specifications. It should be noted that usual modal specification are relevant to express properties on single transition by specifying that a transition shall be a (*necessary*) *must*-transition or a (*admissible*) *may*-transition. However, they do not allow to express requirements on several transitions. For instance, expressing the property p_1 using usual modal specification allows us to specify that transitions of *SubA* and *SubB* shall be *may*-transitions. Nevertheless, such formula does not ensure that *SubA* or *SubB* has to occur in an exclusive manner, and specifying some transitions as *must*-transition cannot tackle this imprecision. This kind of properties highlights the expressive limitation of usual modal specifications and has motivated the definition of extended modal specifications as introduced in Section 3.1.1 page 44. Indeed, by expressing behavioural properties over several transitions and their causalities, extended modal specifications are able to effectively describe such properties.

5.1.2/ Question and Answer Portal

The second study case concerns a business process workflow of a *Question and Answer* portal, which is a part of a proprietary issue tracking system considered in Section 5.1.1.

General Description

This *Question and Answer* portal allows company's customers to ask questions that are then answered by the company's sellers. To use the system, new users (e.g., new employee, new client) have to be registered. Three types of users can log-in: clients, sellers and administrators. The clients can ask questions. These questions are then answered by sellers. Once the answer to a question has been validated by the client who asked it, the administrator archives the question. An execution of the workflow is complete once all users have been logged-out and unregistered.

Modelling

We present one of the several refinements of the workflow modelled by a coloured workflow net. For clarity, this coloured workflow net is described by several coloured *sub*-workflow nets (Figures 5.2, 5.3(a), 5.3(b), 5.3(c), and 5.3(d)). In these figures, places with the same name denote the same entities: for instance, place *HomeQA* in Figure 5.2 and Figure 5.3(a) denotes the same place.

In this refinement, the following three colours are considered:

- $U = \{u_1, .., u_t\}$, a set of t user names representing the different users of the system;
- $R = \{client, seller, admin\}$, a set of roles, which are assigned to users;
- $Q = \{unanswered, answered, validated\}$, a set of question statuses.

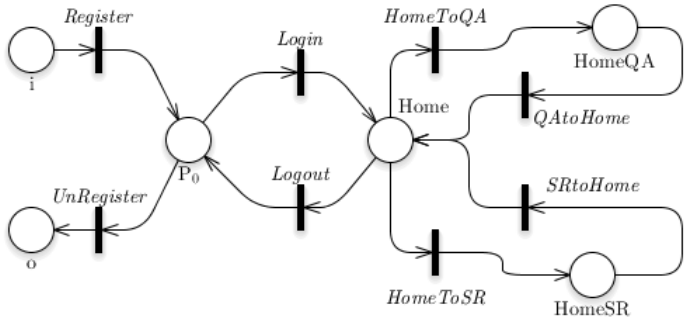
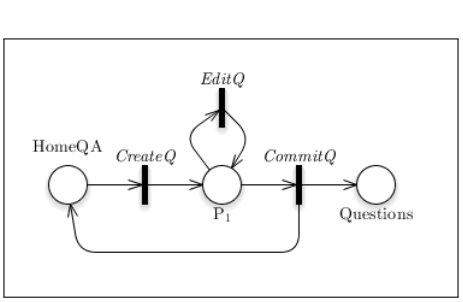


Figure 5.2: Login and navigation coloured *sub*-workflow net

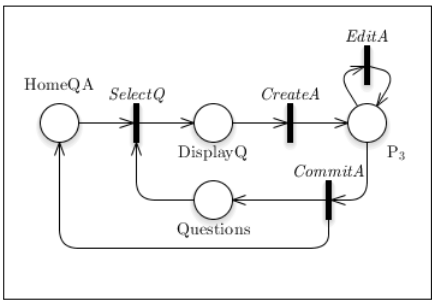
Table 5.2 shows the colours associated with places of the *Question and Answer* coloured workflow net, and Tab. 5.3 shows the colours, inputs, outputs and guards ($u, u_1, u_2 \in U, r \in R, q \in Q$).

Colours	Places
U	i, o
$U \times R$	$P_0, Home, HomeQA, HomeSR$
$U \times R \times Q \times U$	$DisplayQ, DisplayA, P_1, P_3$
$Q \times U$	$Questions$

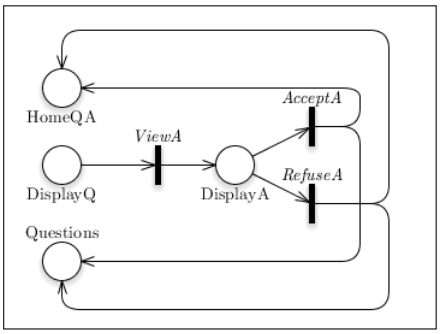
Table 5.2: Colours of *Question and Answer* coloured workflow net's places



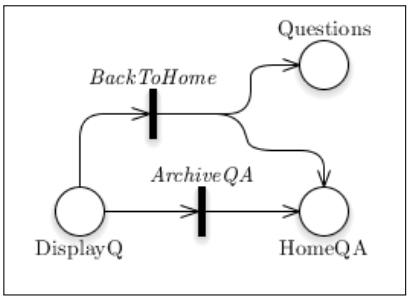
(a) Create Question



(b) Answer Question



(c) (Un)Valid Answer



(d) Archive Question

Figure 5.3: *sub*-CWF-nets of the *Question and Answer* CWF-net

Transition	Colours	Inputs	Outputs	Guard
$Register$	$U \times R$	u	(u, r)	$True$
$UnRegister$	$U \times R$	(u, r)	u	$True$
$Login, Logout$	$U \times R$	(u, r)	(u, r)	$True$
$HomeToQA$	$U \times R$	(u, r)	(u, r)	$True$
$QAtoHome$	$U \times R$	(u, r)	(u, r)	$True$
$HomeToSR$	$U \times R$	(u, r)	(u, r)	$True$
$SRtoHome$	$U \times R$	(u, r)	(u, r)	$True$
$CreateQ$	$U \times R$	(u, r)	$(u, r,' unanswered', u)$	$r =' client'$
$EditQ$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r, q, u_2)	$True$
$CommitQ$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r) and (q, u_2)	$True$
$SelectQ$	$U \times R \times Q \times U$	(u_1, r) and (q, u_2)	(u, r, q, u)	$True$
$CreateA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	$(u_1, r,' answered', u_2)$	$r =' seller' \wedge q =' unanswered'$
$EditA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r, q, u_2)	$True$
$CommitA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r) and (q, u_2)	$True$
$ViewA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r, q, u_2)	$u_1 = u_2 \wedge q =' answered'$
$AcceptA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r) and $('validated', u_2)$	$True$
$RefuseA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r) and $('unanswered', u_2)$	$True$
$ArchiveQA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r)	$r =' admin' \wedge q =' validated'$
$BackToHome$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r) and (q, u_2)	$True$

Table 5.3: Colours, inputs, outputs, and guards of *Question and Answer* coloured workflow net's transitions

An execution of the *Question and Answer* coloured workflow net begins with at least three users (one client, one seller, and one administrator) and is complete once all users have been logged-out and unregistered.

Example 20: Example of an execution of the *Question and Answer* coloured workflow net

To illustrate how this coloured workflow net works, let us consider the following execution with u_1, u_2 and u_3 as initial marking: each user is registered, then he/she logs in and navigates to the *QA*'s Home (Figure 5.2):

- $Register(u_1, client)$, $Login(u_1, client)$, $HomeToQA(u_1, client)$
- $Register(u_2, seller)$, $Login(u_2, seller)$, $HomeToQA(u_2, seller)$
- $Register(u_3, admin)$, $Login(u_3, admin)$, $HomeToQA(u_3, admin)$

The client creates a new question (Figure 5.3(a)):

- $CreateQ(u_1, client)$, $CommitQ(u_1, client, unanswered, u_1)$

The seller selects the question and the answer (Figure 5.3(b)):

- $SelectQ(u_2, seller, unanswered, u_1)$, $CreateA(u_2, seller, unanswered, u_1)$
- $CommitA(u_2, seller, answered, u_1)$

The client selects the question, reads the answer and validates (Figure 5.3(c)):

- $SelectQ(u_1, client, answered, u_1)$, $ViewA(u_1, client, answered, u_1)$
- $AcceptA(u_1, client, answered, u_1)$

The administrator selects the question and archives it (Figure 5.3(d)):

- $SelectQ(u_3, admin, validated, u_1)$,$ArchiveQA(u_3, admin, validated, u_1)$

The users navigate to Home and then log-out and are unregistered (Figure 5.2):

- $QAtoHome(u_1, client)$, $Logout(u_1, client)$ $UnRegister(u_1, client)$
- $QAtoHome(u_3, seller)$, $Logout(u_3, seller)$ $UnRegister(u_2, seller)$
- $QAtoHome(u_3, admin)$, $Logout(u_3, admin)$ $UnRegister(u_3, admin)$

Requirements

Regarding this business process, the goal is to verify, at the specification or design stage of the development, some required behavioural properties derived from textual requirements and business analyst expertise.

The considered behavioural properties, denoted p_i for later references are presented in Table 5.4.

Let us, one more time, emphasize that these properties could not be expressed by usual modal specifications as they involve several transitions as well as constraint over data and the initial states. This is why they have also motivated the definition of extended modal specifications as introduced in Section 3.2.2 page 59.

#	Property
p_7	All users must register
p_8	An admin may view an answer
p_9	A client may create a question and refuse the answer
p_{10}	When a client asks a question it must be answered and archived
p_{11}	A client must not answer a question
p_{12}	There may be an user who asks a question while another does not
p_{13}	If there is less than three users, no question is asked
p_{14}	If a question is asked then the system must have registered a client, a seller and an administrator

Table 5.4: *Question and Answer* portal requirements

5.1.3/ Tax Accounting Manager

The third study case deals with a system to manage tax accounting. It is inspired by a real-life system that provides dependable, automated, efficient and integrated services to more than 400 tax municipalities and more than 3000 users.

Description

This tax accounting manager is huge and handles a vast amount of tax processes which can be, at the early stage of development life cycle, described by workflows in order to specify and to verify some behavioural properties of the payout process before going to implementation.

The payout process handles credit claims (i.e. notes issued when the tax department owes money to a taxpayer). It is composed of three sub-processes, namely, the approval process, the payout form generation process, and the payout process.

It proceeds as follows. Firstly, a credit claim must be approved. Its approbation may need two levels of approval, which can be either manual or automatic depending on the properties of the claim. Secondly, a suggested payout form is generated. It may then be corrected by a tax processor and, if corrected, it must be verified again. Once approved the payout form is sent to the bank which responds with a receipt. The receipt is then handled by a tax processor. Although this payout process is only a small portion of the whole system, its workflow is a representative example as it illustrates a real-life process of high importance.

Modelling

Figure 5.4 presents one of the first refinements of this payout process modelled as a coloured workflow Petri net composed of 10 places and 16 transitions.

The approval process workflow must be refined to specify the property that any credit claim must be approved before being treated. To this end, two colors are defined:

- $C_1 = \{lvl0, lvl1, lvl2, lvl12, approved\}$, a set of levels required to approve a claim
- $C_2 = C_1 \times C_1$, a set of 2-uple whose first element represents the level required to approve the claim, and whose second element represents the actual level of the claim approval

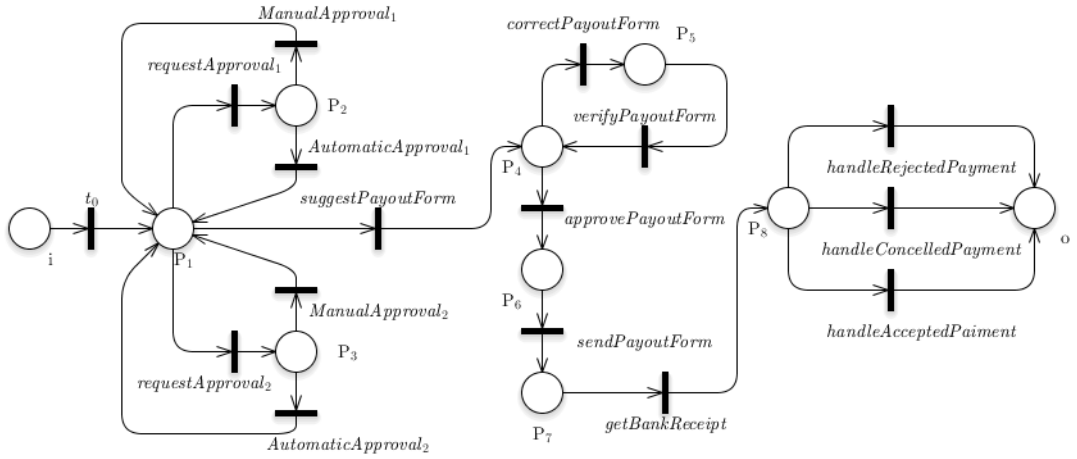


Figure 5.4: *Payout Process* of the *Tax Accounting Manager* study case

The colour C_2 is assigned to places i, P_1, P_2 and P_3, as well as to transitions $requestApproval_{1/2}$, $ManualApproval_{1/2}$, $AutomaticApproval_{1/2}$ and $suggestPayoutForm$. A form is suggested if and only if the credit claim has been approved at the required level(s) as described in Tab. 5.5.

Transition	Instance	Pre	Post
$requestApproval_1$	$lvl1$	$(lvl1, lvl0)$	$(lvl1, lvl0)$
	$lvl12$	$(lvl12, lvl0)$	$(lvl12, lvl0)$
$requestApproval_2$	$lvl2$	$(lvl2, lvl0)$	$(lvl2, lvl0)$
	$lvl12$	$(lvl12, lvl1)$	$(lvl12, lvl1)$
$(Manual/Automatic)Approval_1$	$lvl1$	$(lvl1, lvl0)$	$(lvl1, approved)$
$ManualApproval_1$	$lvl12$	$(lvl12, lvl0)$	$(lvl12, lvl1)$
$(Manual/Automatic)Approval_2$	$lvl2$	$(lvl2, lvl0)$	$(lvl2, approved)$
	$lvl12$	$(lvl12, lvl1)$	$(lvl12, approved)$
$suggestPayoutForm$	lvl_x	$(lvl_x, approved)$	$(lvl_x, approved)$

Table 5.5: Function associated with transitions of the *Payout Process* (Figure 5.4)

Moreover, some properties of interest for this system require performance measures of the mean cost of executions. To this end, weights representing the mean cost of the activities are associated with transitions to constitute a weighted transitions Petri net (Section 2.1.5.3 page 21). This defines a price weight function, that assigns to any transition instance an amount of units needed to perform it. It is a weight function defined in the framework where C is the set of integers, and $\mathfrak{C}$ is the summation of elements of a set. The total weight cost of a correct execution represents the total amount of units needed to perform it. In this study case context, the units in question are unit of money which represents the cost of the considered activity. The total weight of an execution represents the amount of money needed to perform it. Table 5.6 gives the mean cost in tenth of a Euro associated with transitions of the payout process. We suppose this mean cost uniform for all instances of the same transition.

We observe that automated treatments have a low mean cost while other activities, like manual approval and correction, have a significantly higher mean cost.

Transition	Weight	Transition	Weight
$requestApproval_1$	10	$requestApproval_2$	10
$AutomaticApproval_1$	10	$AutomaticApproval_2$	10
$ManualApproval_1$	400	$ManualApproval_2$	500
$suggestPayoutForm$	10	$approvePayoutForm$	120
$correctPayoutForm$	100	$verifyPayoutForm$	80
$sendPayoutForm$	50	$getBankReceipt$	10
$handleRejectedPayment$	80	$handleRejectedPayment$	60
$handleAcceptedPayment$	30		

Table 5.6: Mean costs associated with transitions of the *Payout Process*

Requirements

This payout process is a component of a larger process, which requires valid activities with respect to a specification. Notably, under certain conditions, some activities may be admissible, while some shall be necessary.

Regarding this business process, the goal is to verify, at the specification or design stage of the development, some required behavioural properties, derived from textual requirements and analyst expertise. Denoted p_i for later references, the properties presented in Table 5.7 are considered.

#	Property
p_{15}	credits claims must be approved
p_{16}	a payout form may be corrected
p_{17}	if a payout form is corrected, it must be verified
p_{18}	a payout form may be corrected three times
p_{19}	getting a receipt from the bank is necessary
p_{20}	any claim that requires two levels of approval requires a manual approval level 1
p_{21}	the treatment of a credit claim may have a mean cost of less than 30 Euros
p_{22}	When restricting to three the number of corrections of a payout form, the cost of a credits claims treatment must not exceed a mean cost of 200 Euros

Table 5.7: *Payout Process* requirements

We can see that those properties all express either admissible or necessary behaviours. To ensure that the specified business process model verifies this kind of business rules, there is a need to express and assess them using modal specifications. In the context of Petri net based workflow modelling, usual modal specifications are relevant to express properties on a single transition, i.e an activity, by specifying that a transition shall be a (*necessary*) *must*-transition or a (*admissible*) *may*-transition. However, they do not allow neither expressing requirements on several transitions, nor the requirements conditioned by characteristics on the treated cases, nor the requirements containing performance indicators of the process executions. For instance, the property p_{17} involves distinct activities, its definition within usual modal specification over a single transition is not possible. We can also see that the property p_{20} has a condition on the credit claims characteristics, whereas the properties p_{21} to p_{22} have restrictions on cost performance characteristics of the process executions.

However, like the properties of the previous study case covering the *Question and Answer* portal, these properties can be expressed by the extended modal specifications framework presented in Section 3.2.2 page 59. Moreover, they have motivated its definition over workflows described as coloured workflow nets extended by quantitative performance specification.

5.2/ Tool Chain Implementation

This section describes the tool chain developed to experimentally validate this thesis proposals, and notably illustrates its use on the three study cases presented in the previous section.

The first part of this section presents an extended modal specification verifier implementation based on the modelling approach described in Chapter 3 page 43. The second part presents an implementation of a reduction tool preserving generalised soundness of workflow nets based on the reduction rules described in Section 4.1 page 68. This reduction tool also enables the user to preserve the (in)validity of a given modal specification interpreted as a *must*-formula or a *may*-formula following the approach described in Section 4.3 page 79.

5.2.1/ Modal Specification Verifier

The approach proposed in Chapter 3 page 43 has been fully automated, allowing practitioners, at any stage of the workflow design and validation, to verify modal properties using an integrated tool chain.

Figure 5.5 depicts the global architecture of the modal verification tool chain, which enables to verify modal specifications using either CLP or SMT resolution methods.

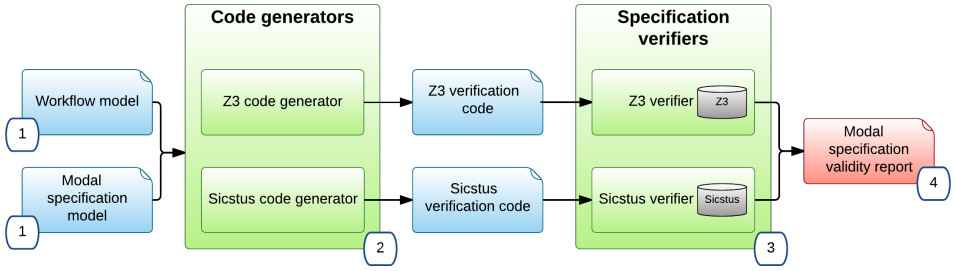


Figure 5.5: Modal specification verification tool chain

First, a workflow net and an extended modal specification formula are given as inputs to the verification software (1). Both models conform to an ad-hoc standard, i.e. a dedicated meta-model, so that all information needed by the verification tool is provided. Note that this implementation also supports input workflow nets model formatted by the PNML standard [Hillah et al., 2010] – a generic Petri nets XML format – allowing the use of workflow nets exported from a third party software (e.g., Yasper [van Hee et al., 2006a], PIPE [Bonet et al., 2007]).

We have also developed a graphical coloured workflow nets editor created within the Sirius framework[2], which is an EMF-based open source project to create customized graphical modelling workbench by leveraging Eclipse Modelling technologies. Basically, it provides a generic workbench for model-based architecture engineering that could be easily tailored to fit the specific needs of a given Domain Specific Language, e.g., coloured workflow nets in our context. Hence

[2]http://projects.eclipse.org/projects/modeling.sirius

the developed coloured workflow nets editor allows producing an XML file, corresponding to the designed coloured workflow net models, that can be used as input file of the tool chain.

Afterwards, the two code generators use the information provided by the input workflow and the input modal specification (1) to generate the code corresponding to the constraint system defined in Section 3.1.3 page 53. Such codes are expressed in the input format of the targeted constraint solver (2): SMT-Lib for Z3 [De Moura et al., 2008] version 4.4.0, an SMT solver, and SICStus Prolog [Carlsson et al., 2012] version 4.3.2, a CLP solver (that includes a CLP(FD) library).

Both solvers were selected for their proven efficiency: Z3 finished first during the 2014 SMT-COMP challenge [Barrett et al., 2005] for solving non-linear arithmetic problems and SICStus obtained the third place during the 2014 MiniZinc challenge [Stuckey et al., 2014].

Next, to verify a *may*-formula (resp. a *must*-formula) m, the tool first checks if there exists a solution of the over-approximation, given by the constraint system $\mathcal{Q}(N, M_i, M_o)$ (Definition 37 page 49), such that the modelled execution satisfies (resp. does not satisfy) m. If such an execution exists, it then tries to find an execution of the under-approximation, given by the constraint system $\mathcal{V}(N, k, m)$ (resp. $\mathcal{V}(N, K, \not m)$) of the Definition 40 page 53.

The function checking the validity of a *must*-formula is given by Algorithm 5. In this algorithm, $SAT(CS)$ is defined as the boolean function returning the satisfiability of the constraint system CS and $GETMODEL(CS)$ is a function returning a valuation function satisfying the constraint system CS. This algorithm returns the K-bounded validity of a given modal formula m. To cope with the complexity raised by K_{max}, K can be fixed to a manageable value. Nevertheless, when fixing K to K_{max} (or greater than K_{max}), the algorithm enables to decide the unbounded validity of the *must*-formula m. The results introduced in Chapter 3 ensure its soundness and completeness.

Finally, once the verification algorithm has been applied through queries to the related solver with the generated code to determine the validity of the modal specification (3), a report is produced (4). This report states the verdict about the validity of the modal specifications as well as the verification time.

Let us point out that, using SICStus, no particular labelling heuristic was found to significantly improve results. Therefore all the experiments have been conducted with the default ones ([`leftmost`, `step`, `up`]). However, using Z3, since the SMT tactic improved all results, this strategy has been systematically used.

The two following subsections respectively illustrate the code generation process (2) and discuss the issue of representing infinite domains.

On Code Generation

To illustrate the code generation process within this tool chain, let us consider the ordinary workflow net $N = (P, T, F)$ depicted in Figure 5.6.

Figures 5.7 and 5.8 illustrate the encoding used to model the constraints seen in Chapter 3 respectively for the SMT-Lib and Prolog language.

Let $n \in P \cup T$, we use the following conventions: `An` stands for $M_a(n)$, `Bn` for $M_b(n)$, `Pn` for $\nu(n)$ when n is a place, `Tn` stands for $\nu(n)$ when n is a transition, and `Xn` stands for $\xi(n)$, `M1n`, `M2n` and `M3n` stand for $M_1(n)$, $M_2(n)$ and $M_3(n)$. For example, `Tt2` and `M2p1` respectively denote $\nu(t2)$ and $M_2(p1)$.

The *initialMarking* and *finalMarking* predicates are used to respectively constrain the initial and

Data: N - a WF-net, m - a *must*-formula, K a positive integer
Result: $TRUE$ - $PN \models_{must} m$, $FALSE$ - $PN \not\models_{must} m$
if $SAT(Q(N, M_i, M_o) \wedge C(N, \neg m))$ **then**
  $v = GETMODEL(Q(N, M_i, M_o) \wedge C(N, \neg m))$;
  $k = max(\{v(n) | n \in T\})$;
  if $k == 1$ **then**
    return $FALSE$;
  else
    while $k \leq K$ **do**
      if $SAT(\mathcal{V}(N, k, \neg m))$ **then**
        return $FALSE$;
      else
        $k = k + 1$;
      end
    end
  end
else
  return $TRUE$;
end

Algorithm 5: Algorithm checking the validity of a *must*-formula

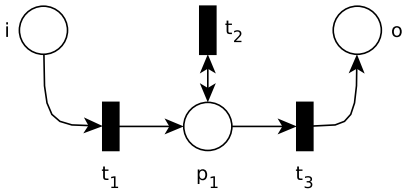


Figure 5.6: An example of workflow net

final marking of a modelled execution. The predicate *formula* defines the constraint imposed by the extended modal specification formula considered: $t_1 \wedge t_2$. The state equation constraint system defined by Definition 35 page 46 is modelled by the *stateEquation* predicate. The subnet of a state equation valuation (Definition 36 page 48) is constructed through the use of the predicate *subnetInit*. The predicate *siphon* models the constraint system of Definition 12 page 49 and is used to determine the presence of a siphon in a workflow net. Based on a solution of the *stateEquation*predicate, the *noShiphon* predicate ensures the absence of siphon within the subnet of the considered state equation valuation. It follows that together the *stateEquation* and *noShiphon* predicates define a predicate *segment* corresponding to the constraint system modelling segment of executions of the considered workflow net as defined by Definition 37 page 49.

```
; Variables declaration here
(define-fun initialMarking ((Ai Int)(Ao Int)(Ap1 Int)) Bool (and (= Ai 1) (= Ao 0) (= Ap1 0)))
(define-fun finalMarking ((Bi Int)(Bo Int)(Bp1 Int)) Bool (and (= Bi 0) (= Bo 1) (= Bp1 0)))
(define-fun stateEquation (
  (Ai Int)(Ao Int)(Ap1 Int)(Bi Int)(Bo Int)(Bp1 Int)(Pi Int)(Po Int)(Pp1 Int)(Tt1 Int)(Tt2 Int)(Tt3 Int)) Bool
  (and
    (>= Ai 0) (>= Ao 0) (>= Ap1 0) (>= Bi 0) (>= Bo 0) (>= Bp1 0)
    (>= Pi 0) (>= Po 0) (>= Pp1 0) (>= Tt1 0) (>= Tt2 0) (>= Tt3 0)
    (= Pi Ai) (= Pi (+ Bi Tt1))
    (= Po (+ Ao Tt3)) (= Po Bo)
    (= Pp1 (+ Ap1 Tt1 Tt2)) (= Pp1 (+ Bp1 Tt2 Tt3))))
(define-fun formula ((Tt1 Int) (Tt2 Int)) Bool (and (> Tt1 0) (> Tt2 0)))
(define-fun noSiphon (
  (Ai Int)(Ao Int)(Ap1 Int)(Bi Int)(Bo Int)(Bp1 Int)(Pi Int)(Po Int)(Pp1 Int)(Tt1 Int)(Tt2 Int)(Tt3 Int)) Bool
  (not (exists ((Xi Int)(Xo Int)(Xp1 Int))
    (and
      (> (+ Xi Xo Xp1) 0)
      (>= Xi 0) (<= Xi 1) (>= Xo 0) (<= Xo 1) (>= Xp1 0) (<= Xp1 1)
      (=> (or (> Ai 0) (> Bi 0) (= Pi 0)) (= Xi 0))
      (=> (or (> Ao 0) (> Bo 0) (= Po 0)) (= Xo 0))
      (=> (or (> Ap1 0) (> Bp1 0) (= Pp1 0)) (= Xp1 0))
      (=> (> Tt1 0) (>= (+ Xi) Xp1)) (=> (> Tt2 0) (>= (+ Xp1) Xp1)) (=> (> Tt3 0) (>= (+ Xp1) Xo))))))
(define-fun segment (
  (Ai Int)(Ao Int)(Ap1 Int)(Bi Int)(Bo Int)(Bp1 Int)(Pi Int)(Po Int)(Pp1 Int)(Tt1 Int)(Tt2 Int)(Tt3 Int)) Bool
  (and
    (stateEquation Ai Ao Ap1 Bi Bo Bp1 Pi Po Pp1 Tt1 Tt2 Tt3)
    (noSiphon Ai Ao Ap1 Bi Bo Bp1 Pi Po Pp1 Tt1 Tt2 Tt3)))
```

Figure 5.7: SMT-Lib representation of a segment of workflow

```
initialMarking([1, 0, 0]).
finalMarking([0, 1, 0]).
stateEquation([Ai, Ao, Ap1], [Bi, Bo, Bp1], [Pi, Po, Pp1], [Tt1, Tt2, Tt3]):-
  domain([Ai, Ao, Ap1, Bi, Bo, Bp1, Pi, Po, Pp1, Tt1, Tt2, Tt3], 0, 10),
  Pi #= Ai, Pi #= Bi + Tt1,
  Po #= Ao + Tt3, Po #= Bo,
  Pp1 #= Ap1 + Tt1 + Tt2, Pp1 #= Bp1 + Tt2 + Tt3.
formula([Tt1, Tt2]):- Tt1 #> 0, Tt2 #> 0.
subnetInit([Ai, Ao, Ap1], [Bi, Bo, Bp1], [Pi, Po, Pp1], [Xi, Xo, Xp1]):-
  domain([Xi, Xo, Xp1], 0, 1),
  (Ai #> 0 #\/ Bi #> 0 #\/ Pi #= 0) #=> Xi #= 0,
  (Ao #> 0 #\/ Bo #> 0 #\/ Po #= 0) #=> Xo #= 0,
  (Ap1 #> 0 #\/ Bp1 #> 0 #\/ Pp1 #= 0) #=> Xp1 #= 0.
siphon([Tt1, Tt2, Tt3], [Xi, Xo, Xp1]):-
  Xi + Xo + Xp1 #> 0,
  Tt1 #> 0 #=> Xi #>= Xp1, Tt2 #> 0 #=> Xp1 #>= Xp1, Tt3 #> 0 #=> Xp1 #>= Xo,
  labeling([leftmost, step, up], [Xi, Xo, Xp1]).
noSiphon([Ai, Ao, Ap1], [Bi, Bo, Bp1], [Pi, Po, Pp1], [Tt1, Tt2, Tt3]):-
  subnetInit([Ai, Ao, Ap1], [Bi, Bo, Bp1], [Pi, Po, Pp1], [Xi, Xo, Xp1]),
  labeling([leftmost, step, up], [Tt1, Tt2, Tt3]),
  \+ siphon([Tt1, Tt2, Tt3], [Xi, Xo, Xp1]).
segment([Ai, Ao, Ap1], [Bi, Bo, Bp1], [Pi, Po, Pp1], [Tt1, Tt2, Tt3]):-
  stateEquation([Ai, Ao, Ap1], [Bi, Bo, Bp1], [Pi, Po, Pp1], [Tt1, Tt2, Tt3]),
  noSiphon([Ai, Ao, Ap1], [Bi, Bo, Bp1], [Pi, Po, Pp1], [Tt1, Tt2, Tt3]).
```

Figure 5.8: Prolog representation of a segment of workflow

As illustration, the two input queries given in Figure 5.9 allow to determine, using respectively Z3 and SICStus, whether there exists a correct execution of the workflow given in Figure 5.6 made of three segments such that both `t1` and `t2` are fired. Note here that these inputs correspond to constraint systems modelling executions composed of at most three segments according to Definition 40 page 53.

Z3 input:

```
(assert (initialMarking M1i M1o M1p1))
(assert (finalMarking M3i M3o M3p1))
(assert (formula (+ T1t1 T2t1 T3t1) (+ T1t2 T2t2 T3t2)))
(assert (segment M1i M1o M1p1 M2i M2o M2p1 P1i P1o P1p1 T1t1 T1t2 T1t3))
(assert (segment M2i M2o M2p1 M3i M3o M3p1 P2i P2o P2p1 T2t1 T2t2 T2t3))
(assert (segment M3i M3o M3p1 M4i M4o M4p1 P3i P3o P3p1 T3t1 T3t2 T3t3))
(check-sat-using smt)
(get-model)
```

SICStus input:

```
initialMarking([M1i, M1o, M1p1]),
finalMarking([M4i, M4o, M4p1]),
segment([M1i, M1o, M1p1], [M2i, M2o, M2p1], [P1i, P1o, P1p1], [T1t1, T1t2, T1t3]),
segment([M2i, M2o, M2p1], [M3i, M3o, M3p1], [P2i, P2o, P2p1], [T2t1, T2t2, T2t3]),
segment([M3i, M3o, M3p1], [M4i, M4o, M4p1], [P3i, P3o, P3p1], [T3t1, T3t2, T3t3]),
S1 #= T1t1 + T1t2, S2 #= T2t1 + T2t2, formula([S1, S2]).
```

Figure 5.9: Input queries using respectively Z3 and SICStus

Both solvers give the following interpretation for the three segments:

```
1 M1i = 1, M1o = 0, M1p1 = 0, M2i = 0, M2o = 0, M2p1 = 1, M3i = 0, M3o = 0, M3p1 = 1, M4i = 0, M4o = 1, M4p1 = 0,
2 P1i = 1, P1o = 0, P1p1 = 1, P2i = 0, P2o = 0, P2p1 = 2, P3i = 0, P3o = 1, P3p1 = 1,
3 T1t1 = 1, T1t2 = 0, T1t3 = 0, T2t1 = 0, T2t2 = 1, T2t3 = 0, 3t1 = 0, T3t2 = 0, T3t3 = 1
```

These segments are given in Figure 5.10, starting (resp. ending) in the initial (resp. final) marking where only the input place `i` (resp. output place `o`) is marked.

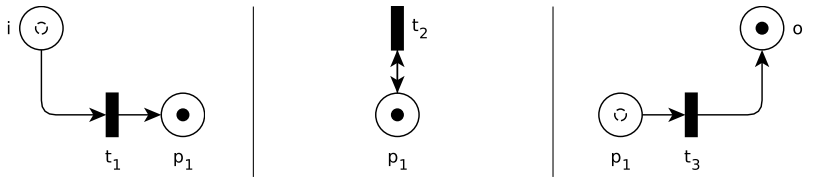


Figure 5.10: The three execution's segments proposed by both solvers

On Infinite Domain Representation

Let us note that modellers often use, in the context of coloured workflow nets development, infinite colours (e.g., strings, integers) to represent data (e.g., usernames of a system, identifiers of files), even if these data are usually not directly manipulated by the control flow. However, coloured workflow nets with infinite colours cannot be directly handled due to the nature of the constraint solvers over finite domains.

Fortunately, abstraction techniques help to tackle the problem entailed by this restriction and can therefore cope with infinite colours. Among well-known abstraction techniques, [Namjoshi et al., 2000] proposes an algorithm to construct a finite state abstract program from a given, possibly infinite, state program (e.g., a coloured workflow net) by means of a syntactic program transformation starting with an initial set of predicates from a specification (e.g., modal specification). This method is shown to be sound (the abstract program is always guaranteed to simulate the original one) and complete (the algorithm can produce a finite simulation-equivalent, resp. bisimulation-equivalent, abstract program if the concrete program has a finite abstraction with respect to simulation, resp. bisimulation, equivalence). On the one hand, in the case of a bisimulation-equivalent abstract program, the abstracted modal specification can be verified using our method, and the (in)validity of the modal specification can be directly inferred. On the other hand, for simulation-equivalent abstract program, only the validity of a *may*-formula and the invalidity of a *must*-formula can be inferred.

To handle infinite colours, another approach is to consider only a finite number of data of an infinite colour according to control-flow selection criteria (e.g., decision or condition coverage) [Vilkomir et al., 2001]. However, this approach is not complete, it only provides a certain level of confidence in the validity or the invalidity of modal specifications depending on the applied selection criterion.

5.2.2/ Reduction Tool

This section presents an implementation of a reduction tool preserving generalised soundness of workflow nets based on the reduction rules described in Section 4.1 page 68 . It notably implements the procedure described by Algorithm 2 of Section 4.2 and has also the ability to reduce workflow nets while preserving the (in)validity of a given modal specification interpreted as a *must*-formula or a *may*-formula by following the approach described in Section 4.3 page 79.

This tool, called *Hadara-AdSimul-Red*, is part of a dedicated open source workflow nets analysis tool suite, called *Hadara-AdSimul*. It has been developed during this thesis to conduct experimental work. This tool (including examples and source code) is available on Github[3].

Figure 5.11 depicts the global architecture of this tool.

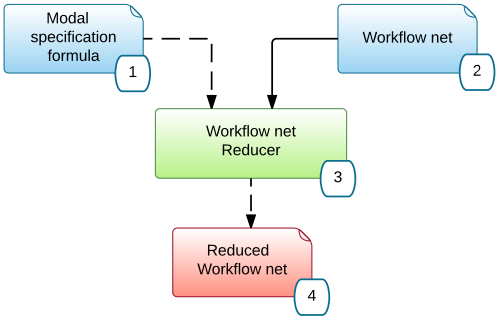


Figure 5.11: Workflow nets reduction tool

This tool takes as input a workflow net (2), saved as an XML file and conform to an ad-hoc and proprietary standard – a dedicated meta-model – or to the PNML standard [Hillah et al., 2010] – a generic Petri nets XML format – so that third party editor can be used (e.g., Yasper [van Hee et al., 2006a], PIPE [Bonet et al., 2007]). It also, possibly, takes as input a modal specification formula (1) conform to an ad-hoc and proprietary XML standard.

The workflow net reducer (3) then tries to apply any of the six reduction rules presented in Section 4.1 page 68 to the input workflow net until a fix-point (i.e. none of the reduction rules can be applied) is reached. If an input modal specification formula has been provided, it instead applies the six restricted reduction rules given in Section 4.3 page 79.

Once the computation is completed, whenever the generalised soundness verification is inconclusive (i.e. the input workflow net could not be completely reduced) or when an input modal specification formula has been provided, this tool produces a reduced workflow net (4) which can then be further analysed. Finally, this tool provides a result report containing the status of the verification of the generalised soundness and metrics about the execution time and the number of times the rules have been applied.

In order to ease and foster its use, this tool features a web interface available online[4].

Figure 5.12 shows, as an example, the web interface output of the execution of *Hadara-AdSimul-Red* when applied to an industrial workflow net in order to determine its generalised soundness. Information and graphic representations of the original and reduced workflow nets can respectively be seen at the left and right of a central frame displaying the required reduction time, the status of the generalised soundness verification as well as the reduction factor obtained.

[3] https://github.com/LoW12/Hadara-AdSimul
[4] http://www.adsimul.com/

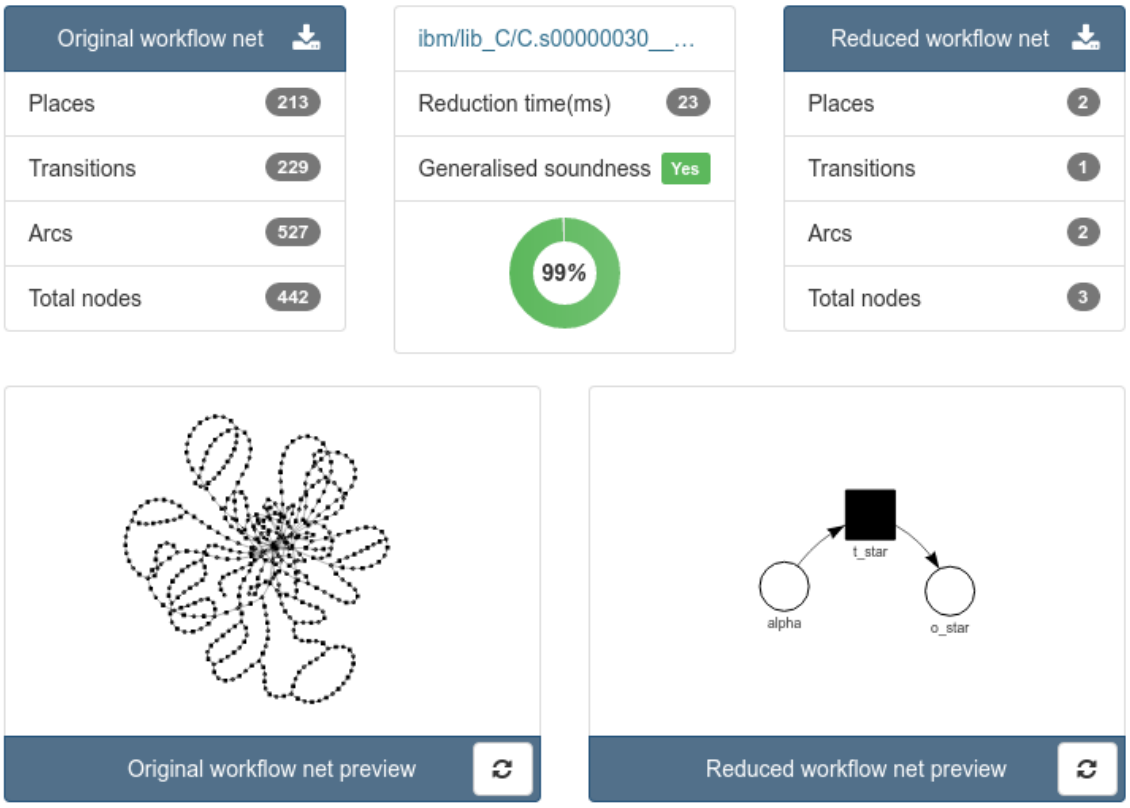


Figure 5.12: Example of the output of the execution of *Hadara-AdSimul-Red*

The following two sections present experimental results obtained by using the tools introduced in this section and the previous. The first one exposes and discusses results obtained over the study cases presented in Section 5.1. The second section presents and discusses, in a first step, an experimental evaluation of the scalability of the proposed extended modal specifications and supporting tool chain over generated benchmarks. In a second step, it introduces an experimental evaluation of the effectiveness and scalability of the reduction approaches implemented by the tool described in this section.

5.3/ Study cases results

For each study case the modal specifications of interest presented in Section 5.1 have been verified using the dedicated modal specification verification tool presented in Section 5.2. The obtained results are presented in this section. Additionally, this section is concluded by a synthesis of the lessons learnt from these results.

Issue Tracking System

We consider the issue tracking system used to manage bugs and issues requested by the customers of a tool provider company. In Section 5.1.1, this study case is presented and six required behavioural properties, denoted $p_1, .., p_6$, are introduced in Table 5.1 page 91. These behavioural properties have been expressed using extended modal specifications (Section 3.1.1 page 44).

Table 5.8 shows the modal specification formulae associated with each behavioural property as

well as the experimental results obtained on this industrial example. The modal *may*-formula (resp. *must*-formula) m associated with each property is specified, and the result of the computation is given by its final result as well as the internal evaluation of the over-approximation $\varphi = Q(N, M_i, M_o) \wedge C(N, m)$ (resp. $\varphi = Q(N, M_i, M_o) \wedge C(N, \neg m)$), the constraint system defined in Definition 37 page 49. The input K and the corresponding computed value of $\phi(K) = \mathcal{V}(N, K, m)$ (resp. $\phi(K) = \mathcal{V}(N, K, \neg m)$), the constraint system defined in Definition 40 page 53, are also stated when the algorithm cannot conclude without this bound. The properties p_1 to p_6 are representa-

#	Formula	φ	K	$\phi(K)$	Result
p_1	$ITS \models_{must} (SubA \wedge \neg SubA) \vee (SubB \wedge \neg SubA)$	TRUE	-	-	TRUE
p_2	$ITS \models_{must} SR_SelectData \Rightarrow Login$	TRUE	-	-	TRUE
p_3	$ITS \models_{must} SR_CreateCRITSIT \Rightarrow (VandD \vee Update \vee Closure)$	TRUE	-	-	TRUE
p_4	$ITS \models_{may} SR_CreateCRITSIT \Rightarrow (Update \wedge Closure)$	FALSE	-	-	FALSE
p_5	$ITS \models_{may} SR_UpgradeToCRITSIT$	TRUE	1	FALSE	FALSE
			6	TRUE	TRUE
p_6	$ITS \models_{must} Login \Rightarrow Logout$	FALSE	1	FALSE	FALSE

Table 5.8: *Issue Tracking System*: Experimentation results

tive of the kind of properties that have to be verified by engineers when they design the business process to be implemented. Moreover, these properties are sufficiently clear without a complete description of the workflow and enable to show all possible outcomes of our approach.

We observe that on the one hand, when verifying *must*-formulae that are satisfied by the considered workflow net (see p_1, p_2 and p_3), or *may*-formulae that are not satisfied by the workflow net (see p_4), the over-approximation proposed in Definition 37 is usually sufficient to conclude. On the other hand, when verifying *may*-formulae that are satisfied by the workflow net (see p_5), or *must*-formulae that are not satisfied by the workflow net (see p_6), the decomposition into K segments is needed. We empirically demonstrate that this decomposition is very effective since values of K_{max} are usually moderate ($K_{max} = 6$ in the case of p_5, less than 10 with all the experimentations on this case-study). We can also notice the definitive invalidity of p_6 (a user can exit the current session without logout), which enabled to highlight an ambiguity in the textual requirements.

On the basis of the experiments performed on this study case and related results, we can conclude that the proposed method is feasible and effective. Furthermore, regarding efficiency, it should be noted that the developed tool is able to conclude about the (in)validity of the studied properties in a very short time (less than a second).

Question and Answer Portal

We now consider the business process workflow of a *Question and Answer* portal. In Section 5.1.2 page 91, this study case is presented and eight required behavioural properties, denoted $p_7, .., p_{14}$, are introduced in Table 5.4 page 95. These behavioural properties have been expressed using abstract extended modal specifications (Section 3.2.2 page 59) considering both initial states and transition instances.

Table 5.9 shows the abstract modal formula associated with each behavioural property as well as the experimental results obtained on this industrial example. The modal *may*-formula (resp. *must*-formula) m associated with each property is specified, and the result of the computation is given by its final result as well as the internal evaluation of the over-approximation $\varphi = Q_a(N, M_i, M_o) \wedge C(N, m)$ (resp. $\varphi = Q_a(N, M_i, M_o) \wedge C(N, \neg m)$). The input K and the corresponding computed value of $\phi(K) = \mathcal{V}_a(N, K, m)$ (resp. $\phi(K) = \mathcal{V}_a(N, K, \neg m)$) are also stated when it makes sense, i.e. when the algorithm cannot conclude without this bound.

#	Formula	φ	K	$\phi(K)$	Result
p_7	$QA \models_{must} \langle true, true, Register[u = u_1] \wedge .. \wedge Register[u = u_t] \rangle$	TRUE	-	-	TRUE
p_8	$QA \models_{may} \langle true, true, ViewA[r = admin] \rangle$	FALSE	-	-	FALSE
p_9	$QA \models_{may} \langle true, true, CreateQ[r = client, u = u_x] \wedge Refuse A[r = client, u_1 = u_x] \rangle$	TRUE	5	FALSE	-
			7	TRUE	TRUE
p_{10}	$QA \models_{must} \langle true, true, CommitQ[u_2 = u_x] \Rightarrow CommitA[u_2 = u_x] \wedge ArchiveQA[u_2 = u_x] \rangle$	TRUE	-	-	TRUE
p_{11}	$QA \models_{must} \langle true, true, \neg CreateA[r = client] \rangle$	TRUE	-	-	TRUE
p_{12}	$QA \models_{may} \langle nbUsers > 3, true, CreateQ[u = u_x] \wedge \neg CreateQ[u = u_y] \rangle$	TRUE	12	TRUE	TRUE
p_{13}	$QA \models_{must} \langle nbUsers < 3, true, \neg CreateQ[true] \rangle$	TRUE	-	-	TRUE
p_{14}	$QA \models_{must} \langle true, true, CreateQ[true] \Rightarrow (Register[r = client] \wedge Register[r = seller] \wedge Register[r = admin] \rangle$	TRUE	-	-	TRUE

Table 5.9: *Question and Answer Portal*: Experimentation results

Since the properties have initially been defined by the business analysts involved in the project, we assume that they are representative of properties that should be verified by engineers when they design and implement such business processes. Furthermore, the obtained verification results have been shared and discussed with them.

As for the previous study case, on the one hand, we observe that when verifying *must*-formulae that are satisfied by the coloured workflow net (e.g., p_7, p_{10}), or *may*-formulae that are not satisfied by the coloured workflow net (e.g., p_8), the over-approximation is usually sufficient to conclude. On the other hand, when verifying *may*-formulae that are satisfied by the coloured workflow net (e.g., p_9), or *must*-formulae that are not satisfied by the coloured workflow net, the decomposition into K segments is needed. We empirically show that this decomposition is very effective since values of K_{max} are usually moderate ($K_{max} = 12$ for p_6, less than 30 on all the experiments conducted on this case study).

Thanks to the experiments conducted on this study case, we can conclude that the proposed method is effective and efficient over coloured workflow nets, and can therefore gain benefits within business process design and verification. Notably, regarding efficiency, these experiments have highlighted once more that the approach is able to conclude about the (in)validity of the studied properties in a very short time (less than 5 seconds using SICStus Prolog).

Tax Accounting Manager

We consider the business process workflow of a *Tax Accounting Manager*. In Section 5.1.3 page 95, this study case is presented and eight required behavioural properties, denoted $p_{15}, .., p_{22}$, are introduced in Table 5.7. These behavioural properties have been expressed using abstract extended modal specifications (Section 3.2.2 page 59) considering both initial states and transition instances together with performance indicators, a new feature.

Table 5.10 shows the abstract modal formula associated with each behavioural property as well as the experimental results obtained on this industrial example. The modal *may*-formula (resp. *must*-formula) m associated with each property is specified, and the result of the computation is given by its final result as well as the internal evaluation of the over-approximation $\varphi = Q_a(N, M_i, M_o) \wedge C(N, m)$ (resp. $\varphi = Q_a(N, M_i, M_o) \wedge C(N, \neg m)$). The input K and the corresponding computed value of $\phi(K) = \mathcal{V}_a(N, K, m)$ (resp. $\phi(K) = \mathcal{V}_a(N, K, \neg m)$) are also stated when it makes sense, i.e. when the algorithm cannot conclude without this bound. We note here that property p_{22}, which expresses necessities, is transformed into a *may*-formula with the negation of the weight constraint imposed on it. It follows that the result of the validity of this property is the negation of the validity of the *may*-formula.

#	Formula	φ	K	$\phi(K)$	Result
p_{15}	$TAM \models_{must} \langle true, true, AutomaticApproval_1[d > 0] \vee AutomaticApproval_2[d > 0] \vee ManualApproval_1[d > 0] \vee ManualApproval_1[d > 0] \rangle$	TRUE	-	-	TRUE
p_{16}	$TAM \models_{may} \langle true, true, correctPayoutForm[d > 0] \rangle$	TRUE	4	TRUE	TRUE
p_{17}	$TAM \models_{must} \langle true, true, correctPayoutForm[d > 0] \Rightarrow verifyPayoutForm[d > 0] \rangle$	TRUE	-	-	TRUE
p_{18}	$TAM \models_{may} \langle true, true, correctPayoutForm[d = 3] \rangle$	TRUE	7	TRUE	TRUE
p_{19}	$TAM \models_{must} \langle true, true, getBankReceipt[d > 0] \rangle$	TRUE	-	-	TRUE
p_{20}	$TAM \models_{must} \langle Mi(i) > (lvl12, lvl0), true, ManualApproval_1[d > 0] \rangle$	TRUE	-	-	TRUE
p_{21}	$TAM \models_{may} \langle true, true, - \rangle \vert C < 300$	TRUE	4	TRUE	TRUE
p_{22}	$TAM \models_{may} \langle true, true, correctPayoutForm[d < 4] \rangle \vert C > 20000$	FALSE	-	-	FALSE

Table 5.10: *Tax Accounting Manager*: Experimentation results

In a similar way to the precedent study cases, we have experimentally demonstrated that verifying *must*-formulae that are satisfied by the coloured workflow net, or *may*-formulae that are not satisfied by the coloured workflow net, does not usually require segment decomposition. The verification is therefore very efficient as only the over-approximation is required. On the contrary, verifying *may*-formulae that are satisfied by the coloured workflow net (e.g., p_{16}), or *must*-formulae that are not satisfied by the coloured workflow net, does require the decomposition into K segments. Nevertheless, we have empirically shown that this decomposition is also effective and very efficient in the context of coloured workflow nets extended with weight functions since values of K are also moderate (less than 10), and efficiently handled by the constraint solver (solved in less than 5 seconds using SICStus Prolog).

Synthesis

The results gathered over these three study cases allow us to conclude that the presented extended modal specification verification over workflows described as workflow nets possibly extended by data and performance indicators is effective and efficient.

They notably highlight the appropriate use of over-approximations for verifying *must*-formulae that are satisfied by the workflow net, or *may*-formulae that are not satisfied by the workflow net.

Using constraint systems satisfiability to check the validity of extended modal specifications enables the usage of efficient constraint solvers to support the verification of workflow nets against extended modal specifications. Moreover, we showed that this method is still valuable when extending modal specifications with data and/or weights.

Indeed, thanks to the experiments conducted using the dedicated tool chain relying on powerful solvers, we can conclude that the proposed method is suitable and helpful, and can therefore gain benefits within business process design and verification. Furthermore, these experiments permitted to demonstrate that the developed tooling is able to conclude about the (in)validity of the studied properties in a very short time.

The promising results obtained over real-life industrial case studies prompted a more broader evaluation to precisely measure the efficiency of this approach over large scale models. To reach this goal, the next section aims at providing an in-depth evaluation in order to assess the scalability of the proposed modal specification verification method over large benchmarks composed of workflow nets of growing size and complexity.

5.4/ Scalability

In a first step, this section presents and discusses an empirical evaluation of the scalability of the verification of extended modal specifications over workflow nets according to the verification approach defined in Chapter 3 page 43. To this end, a workflow nets generation tool, able to produce workflow nets of growing size and complexity together with (in)valid extended modal specifications, is first introduced. On the basis of such generated workflow nets, an experimental evaluation aiming at assessing the scalability of the verification approach based on such generated benchmarks is then defined. The obtained results are presented and discussed.

In a second step, this section introduces and discusses an experimental evaluation of the effectiveness and scalability of the workflow nets reduction approaches defined in Section 4.2 page 77 and 4.3 page 79.

5.4.1/ Benchmark's Generation Tool

In this section we introduce three tools forming a tool chain able to produce large benchmarks of workflow nets together with their (in)valid extended modal specifications.

The first tool, called *Hadara-AdSimul-Formula-Gen*, produces random extended modal formulae. The second tool, called *Hadara-AdSimul-Formula-To-WF*, produces workflow nets (in)valid with respect to given modal formulae. The third tool, called *Hadara-AdSimul-Expander*, produces larger and more complex workflow nets from given workflow nets while preserving the (in)validity of the modal formulae associated with them. All three tools are part of *Hadara-AdSimul*, a dedicated open source tool suite previously introduced in Section 5.2.2 page 103.

Figure 5.13 depicts the global architecture of the tool chain able to produce large benchmarks of workflow nets together with their (in)valid modal specifications. The elements in the middle line of the figure represent the thee tools above-mentioned, while the elements above and below represent data files.

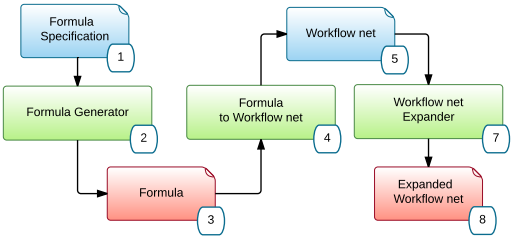


Figure 5.13: Architecture of the generation toolchain

The entry point of the tool chain is an XML file that contains the specifications of the formula to be generated (1): its intended size as well as the probability at which the operator *and*, *or*, and *not* will be used by the Formula Generator (2) to produce a Formula (3) verifying these given specifications. According to these specifications, the Formula Generator (2), starting from a single literal, recursively and randomly (according to the given probability) replaces an existing literal by either its negation, or its conjunction/disjunction with a newly introduced literal. Once the desired formula's size is reached, the produced formula is then saved in an XML file (3).

The next step consists in producing an XML file describing a workflow net (5) satisfying (resp. not satisfying) the modal formula (resp. the negation of the modal formula) of the input formula interpreted as a *must*-formula. This computation is performed using the Formula to Workflow

net tool (4) that maps each operator of the input formula to its corresponding workflow net structure (Section 2.1.6). Finally, the produced workflow net (5) is expanded using the Workflow Expander tool (7). This expansion is achieved by successively applying synthesis rules (inverse of the reduction rules presented in Section 4.1 page 68) which add transitions and/or places while preserving the validity of the modal specification associated. Note that this workflow nets generation approach is similar to the one proposed in [Van Hee et al., 2010]. However the synthesis rules used by our workflow nets generation approach are generalisation of the synthesis rules used in [Van Hee et al., 2010]. Therefore they are able to generate workflow nets exhibiting a broader range of behaviours. The result is an expanded workflow net saved as an XML file (8) satisfying or not the modal formula of the produced formula (3) interpreted as a may-formula or a must-formula. The produced formula (3) and the produced expanded workflow net (8) together form an instance of a benchmark.

Example 21: Example of workflow net generation

To illustrate this process we consider the following example. A *must*-formula of size 5 is generated using the Formula Generator (2): $t_0 \wedge t_1 \wedge (t_2 \vee t_3 \vee t_4)$. The Formula to Workflow net tool (4) is then used to produce a workflow net, depicted in Figure 5.14, satisfying this *must*-formula. This workflow is then extended to the desired size of 50 nodes using the Expander tool (7) while preserving the validity of the considered *must*-formula. The obtained workflow is depicted in Figure 5.15.

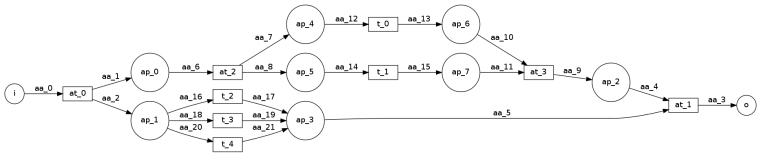


Figure 5.14: A workflow net satisfying the *must*-formula $t_0 \wedge t_1 \wedge (t_2 \vee t_3 \vee t_4)$

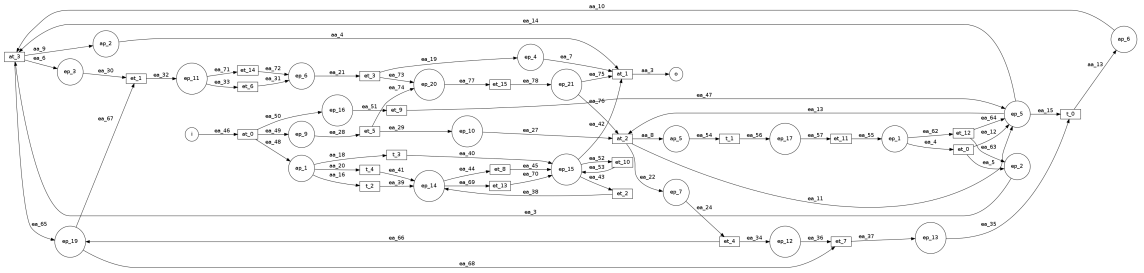


Figure 5.15: The extended workflow obtained from the workflow of Figure 5.14

In this way, the presented tool chain is able to produce large benchmarks of workflow nets together with their (in)valid modal specifications. In the next section, an experimental protocol using this tool chain is designed to experimentally evaluate the scalability of the extended modal specification verification approach presented in Chapter 3 page 43.

5.4.2/ Experimental Evaluation of Modal Specification Verification

In this section, we present an experimental evaluation of the verification of extended modal specifications over workflow nets according to the approach presented in Chapter 3 page 43.

To this end, we first detail the objectives of the proposed experimental evaluation. Second, we define the experimental protocol. Finally, the experimental results obtained according to the defined experimental protocol are presented and discussed.

5.4.2.1/ Objectives

The primary objectives of this experimental evaluation are to experimentally assess the *effectiveness*, *efficiency* and *scalability* of the proposed extended modal specification verification method over workflow nets of growing size and complexity.

In the context of this experimentation, the proposed method is said to be effective over a given instance – a workflow net together with its extended modal specification – if and only if it is able to assign a verdict about the (in)validity of the given modal specification in an *admissible time*. If the proposed method is effective over a given modal specification, its efficiency is its ability to return such a verdict in the least amount of time and memory. It follows that the proposed method is scalable if it is effective and efficient over workflow nets of growing size and complexity.

The secondary objective of this experimental evaluation is to compare the relative efficiency of the proposed method when employing two different constraint resolution approaches: Constraint Logic Programming and Satisfiability Modulo Theories (Section 2.3 page 38).

5.4.2.2/ Experimental Protocol

On the basis of the previously introduced objectives, the following experimental protocol have been designed.

Regarding effectiveness assessment, an admissible time is arbitrary fixed to a reasonable value of 10 minutes (i.e. the time-out of constraint solver queries if fixed to 10 minutes). To evaluate the effectiveness of the proposed method, this protocol thus consists in gathering, the proposed verification approach ability to assign a verdict to the instances of the considered data set in an admissible time of 10 minutes. If a verdict is assigned, the required time is also gathered in order to evaluate the proposed verification approach efficiency.

Furthermore, in order to compare the relative efficiency of the proposed method when employing two different constraint resolution approaches (CLP vs SMT), each instance of the considered data set has to be evaluated once using a CLP constraint solver and once using a SMT solver.

To make conclusion and feedback relevant and credible, and to be able to evaluate the scalability of the extended modal specification verification approach, these experimental results has to be calculated from a broad range of modal specifications and workflow nets. Indeed, the type of modal specifications shall be taken into account because, to conclude about their validity, the verification method may require the computation of the over-approximation of the workflow nets executions or a full decomposition into segments. The proposed experimental protocol thus considers workflow nets of realistic size by evaluating workflow nets of size up to 500 nodes. Moreover, not only the size of the workflow nets is considered but also their complexity by evaluating workflow nets of classes with a growing expressiveness (state machines, marked graphs, free-choice workflow nets and ordinary Petri nets). Additionally, the size of the modal formula to be verified is also important since a larger formula may constrain further the system to be solved. Figure 5.16 summarises the parameters considered and their values taken into account.

- Class of the workflow nets:
  - State machines
  - Marked graphs
  - Free-choice nets
  - Ordinary Petri nets
- Size of the workflow nets:
  - $50 * i$ where $i \in \{1, .., 10\}$
- Type of modal specification:
  - Valid may-formula
  - Invalid may-formula
  - Valid must-formula
  - Invalid must-formula
- Size of the modal formula:
  - 5 and 15 literals

Figure 5.16: Workflow nets generation parameters and their values

For each combination of the listed parameters values, a corresponding modal formula and a workflow net have been randomly generated thanks to the tool chain described in Section 5.4.1. This produced 320 instances of growing size and complexity. Repeating this generation process three time led to the data set considered by this experimental protocol. A total of 960 workflow nets and modal specifications are thus considered in this protocol.

5.4.2.3/ Results and Feedback from Experiments

This section presents the experimental results obtained using the dedicated tools described in Sections 5.2.1 by applying the experimental protocol introduced in the previous section. Let us remind that this tool rely on two constraint solvers: Z3 [De Moura et al., 2008] version 4.4.0, an SMT solver, and SICStus Prolog [Carlsson et al., 2012] version 4.3.2, a CLP solver. For each instances of the data set the results have been gathered using both solvers and all the executions have been computed on a computer featuring an Intel(R) Xeon(R) CPU X5650 @ 2.67GHz.

The obtained results are discussed by distinguishing two different categories of modal specifications: one for may-valid and must-invalid specifications, and one for may-invalid and must-valid specifications. Indeed, using the verification algorithm given in Section 5.2.1, most may-valid and must-invalid modal specifications can be verified by using only an over-approximation of correct executions of the workflow. This over-approximation is less complex than the under-approximation that must very often be computed to verify may-invalid and must-valid modal specifications.

We also categorise the results according to the different classes of workflow nets considered in our experimental protocol. The average execution's times given below has been calculated without considering time-outs. Finally, for clarity, time-outs and singularities have been withdrawn from some plots. The interested reader can however study the complete data sets and results given at https://dx.doi.org/10.6084/m9.figshare.2067156.v3.

Tables 5.11, 5.12, 5.13 and 5.14 summarize the average verification times, number of time-outs as well as the overall appreciation of the results obtained over the studied workflow net classes. The emoticons of the overall appreciation are interpreted as follows:

- 🙂 → Reasonable time, no time-out,
- 😐 → Reasonable time, # time-outs < 50%,
- 🙁 → # time-outs > 50%

Figures 5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23 and 5.24 depict the plots displaying the verification times respectively spent by SICStus and Z3 for each class of workflow nets and types of modal specifications.

The next subsections comment on these obtained results and indicate the most important feedback for each class of workflow nets with regards to the intended objectives of the experimentations, as given in Section 5.4.2.1.

Type	Solver	Avg. t.(ms)	#time-outs	Overall
May-Valid	Z3	346	0	
	SICStus	621	28	
Must-Invalid	Z3	319	0	
	SICStus	788	31	
May-Invalid	Z3	79	0	
	SICStus	77413	52	
Must-Valid	Z3	79	0	
	SICStus	10194	51	

Table 5.11: Metrics over state-machine workflow nets

Type	Solver	Avg. t.(ms)	#time-outs	Overall
May-Valid	Z3	630	0	
	SICStus	776	0	
Must-Invalid	Z3	641	0	
	SICStus	758	0	
May-Invalid	Z3	112	0	
	SICStus	424	0	
Must-Valid	Z3	104	0	
	SICStus	407	0	

Table 5.12: Metrics over marked-graph workflow nets

Type	Solver	Avg. t.(ms)	#time-outs	Overall
May-Valid	Z3	379	0	
	SICStus	787	16	
Must-Invalid	Z3	413	0	
	SICStus	898	14	
May-Invalid	Z3	91	0	
	SICStus	40459	38	
Must-Valid	Z3	89	0	
	SICStus	50566	37	

Table 5.13: Metrics over free-choice workflow nets

Type	Solver	Avg. t.(ms)	#time-outs	Overall
May-Valid	Z3	1258	22	
	SICStus	9010	33	
Must-Invalid	Z3	713	17	
	SICStus	12258	37	
May-Invalid	Z3	108	0	
	SICStus	9489	33	
Must-Valid	Z3	106	0	
	SICStus	5949	37	

Table 5.14: Metrics over ordinary workflow nets

Observation from State-Machine Workflow Nets Verification

May-Valid and Must-Invalid specifications.

Both solvers were able to conclude in a comparable and reasonable time. On average, Z3 execution's times was 332ms whereas SICStus execution's times was 704ms. However, despite good results for both solvers, it should be noted that 49.2% (59/120) of SICStus executions did not finish within 10 minutes, while Z3 did not suffer from any time-outs.

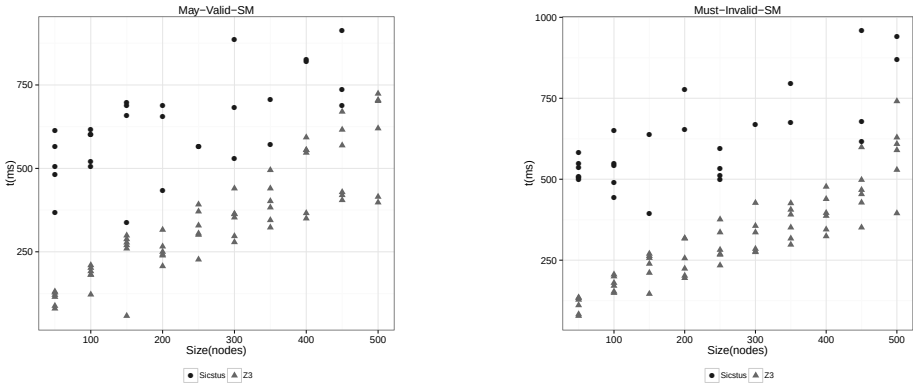


Figure 5.17: State-machine - may-valid and must-invalid modal specifications

Must-valid and May-Invalid specifications.

Both solvers were able to conclude in a reasonable time. On average, Z3 execution's times was 79ms whereas SICStus execution's times was 43803ms. These results clearly show that Z3 performs better than SICStus on this type of modal specifications. Moreover, it should be noted that 85.8% (103/120) of SICStus executions did not finish within 10 minutes, whereas Z3 did not suffer from any time-outs. Indeed, SICStus was not able to conclude about workflow nets of size greater than 250.

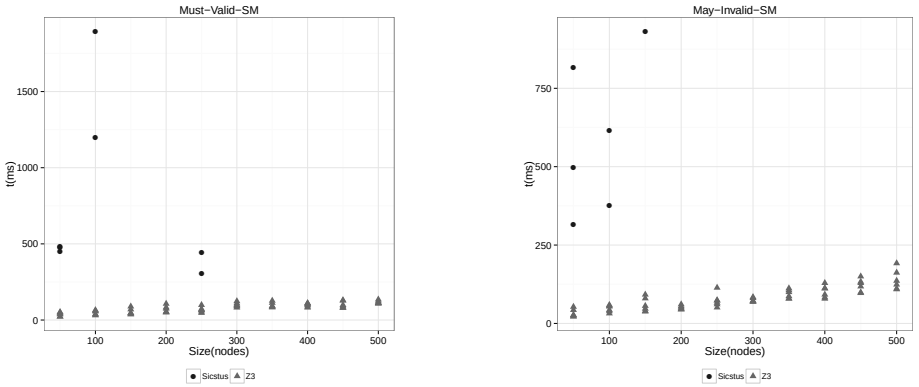


Figure 5.18: State-machine - must-valid and may-invalid modal specifications

Synthesis. Over the class of State-Machines we observe that proposed verification method is effective and efficient using Z3. However, we also observe that when using SICStus the approach is less effective. Indeed, we can observe that SICStus is clearly overwhelmed due to the high number of choice points arising from the structure of state-machine workflow nets of size greater than 100 nodes.

Observation from Marked-Graph Workflow Nets Verification

May-Valid and Must-Invalid specifications.

Both solvers were able to conclude in a reasonable and comparable time. On average, Z3 execution's times was 635ms whereas SICStus execution's times was 767ms. It should also be pointed out that for large sized Marked-Graph workflow nets (greater than 400 nodes) SICStus performs slightly better than Z3.

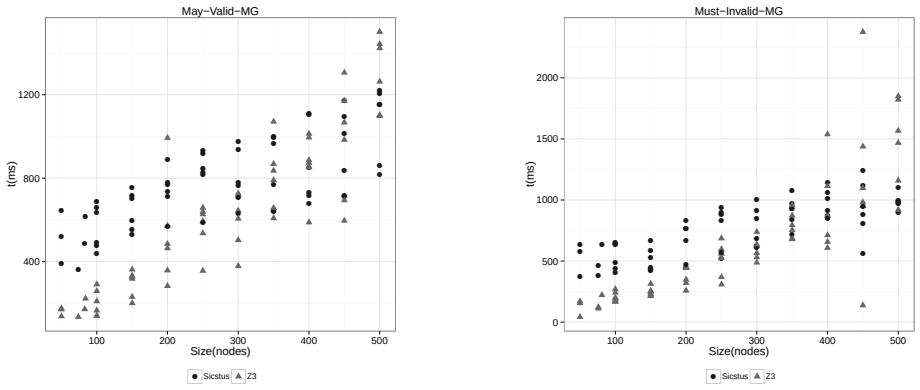


Figure 5.19: Marked-graph - may-valid and must-invalid modal specifications

Must-valid and May-Invalid specifications.

Both solvers were able to conclude in a reasonable and fairly comparable time. On average, Z3 execution's times was 108ms whereas SICStus execution's times was 415ms. Besides, it should be noted that, for this type of modal specifications, Z3 performs slightly better than SICStus.

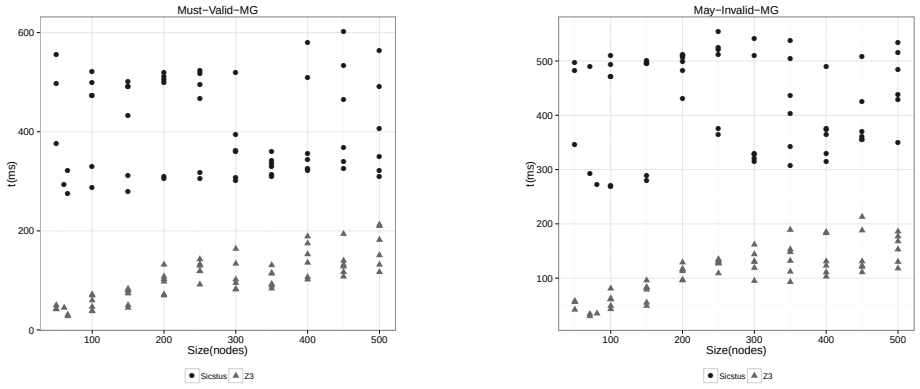


Figure 5.20: Marked-graph - must-valid and may-invalid modal specifications

Synthesis. Over the class of Marked-Graph, we observe the proposed verification approach is effective and efficient using ether SICStus or Z3. We also point out that SICStus performs slightly better when verifying May-Valid and Must-Invalid specifications while Z3 performs slightly better when verifying Must-valid and May-Invalid specifications. A further investigation has shown that, in general, Z3 is more effective than SICStus for the computation of the over-approximation used by the verification method, while SICStus is more effective than Z3 for the computation of the segments needed to conclude whenever the over-approximation is not sufficient.

Observation from Free-Choice Workflow Nets Verification

May-Valid and Must-Invalid specifications.

Both solvers were able to conclude in a reasonable and comparable time. On average, Z3 execution's times was 396ms whereas SICStus execution's times was 842ms. Beyond these conclusive results for both solvers, it is important to underline that 25% (30/120) of SICStus executions did not finish within 10 minutes, whereas Z3 did not suffer from any time-outs.

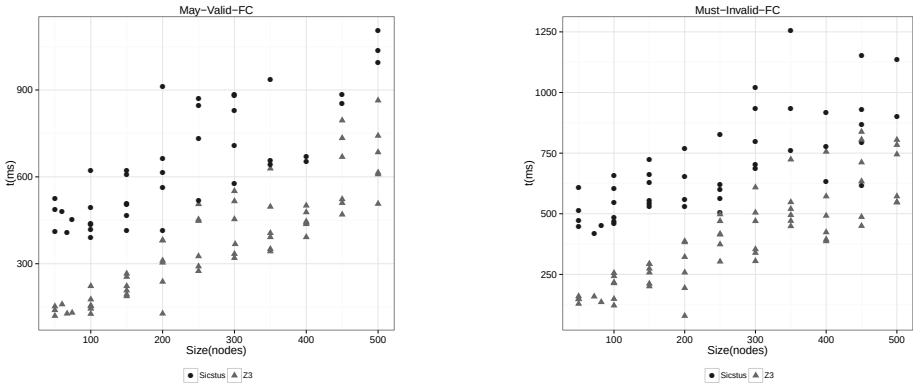


Figure 5.21: Free-choice - may-valid and must-invalid modal specifications

Must-valid and May-Invalid specifications.

Over this type of modal specifications, Z3 clearly performs better than SICStus. On average, Z3 execution's times was 90ms, while SICStus execution's times was 45512ms. We also note that 62.5% (75/120) of SICStus executions did not finish within 10 minutes, whereas Z3 did not suffer from any time-outs. After investigation, these results stem from the fact that the verification of such modal specifications mostly relies on the results of an over-approximation for which Z3 performs far better off.

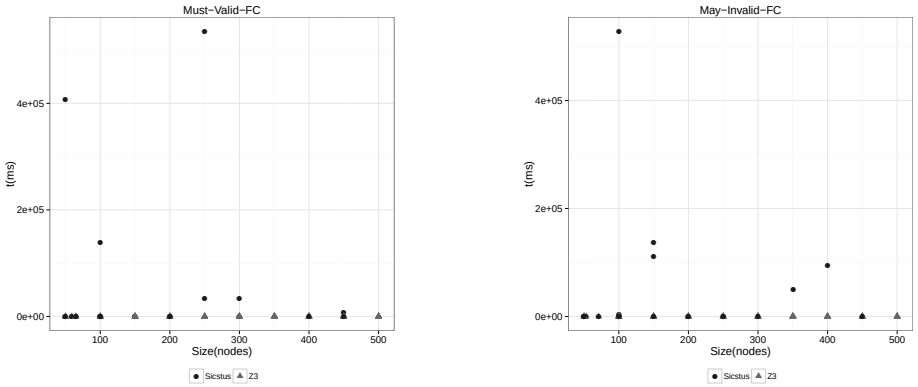


Figure 5.22: Free-choice - must-valid and may-invalid modal specifications

Synthesis. Over the class of Free-Choice workflow nets, we can observe that, using Z3, the proposed verification method is effective and efficient. We also note that the proposed approach is less effective when using SICStus. We thus conclude from these obtained results that the SMT approach seems to be more suited for the verification of modal specifications over Free-Choice workflow nets.

Observation from Ordinary Workflow Nets Verification

May-valid and Must-Invalid specifications.

Both solvers were able to conclude in a reasonable and comparable time. On average, Z3 execution's times was 985ms whereas SICStus execution's times was 10634ms. Besides these results, it should be underlined that 58.3% (70/120) of SICStus executions and that 32.5% (39/120) of Z3 executions did not finish within 10 minutes. However, for each instance, at least one resolution method was indeed able to assign a verdict within 10 minutes.

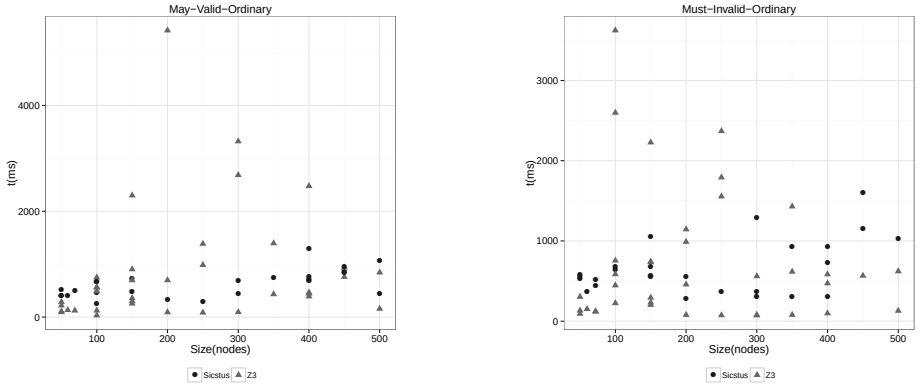


Figure 5.23: Ordinary workflow nets - may-valid and must-invalid modal specifications

Must-valid and May-Invalid specifications.

On average, Z3 execution's times was 107ms whereas SICStus execution's times was 7717ms. Despite these good results for both solvers, it is important to note that 58.3% (70/120) of SICStus executions did not finish within 10 minutes, whereas Z3 did not suffer from any time-outs. As for the previous classes, Z3 indeed performs better than SICStus to compute the over-approximation constraints, which were often sufficient to conclude.

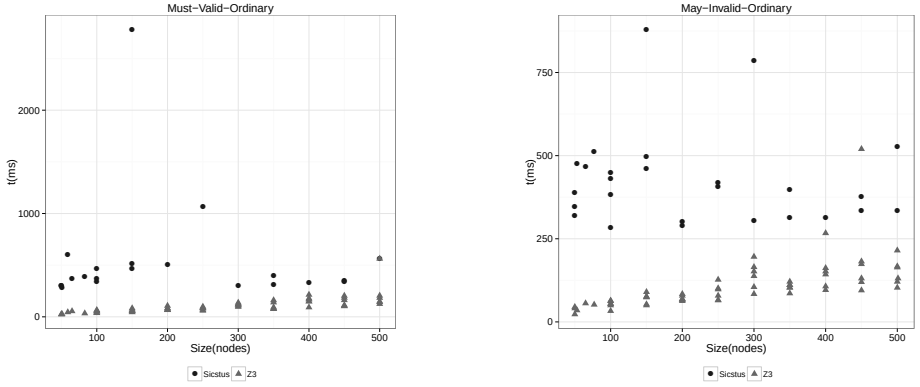


Figure 5.24: Ordinary workflow nets - must-valid and may-invalid modal specifications

Synthesis. Over the class of ordinary workflow nets, we can observe that the proposed verification method is effective and efficient. We also observe that Z3 performs better than SICStus, especially when verifying Must-valid and May-Invalid specifications. We can thus conclude from these results that the SMT approach seems to be more suited for the modal specifications verification over ordinary workflow nets. Moreover, further investigations have pointed out that the limits of the proposed method are reached when considering workflow nets of size greater than 500 nodes.

The next section summarizes the lessons learned and the benefits noticed from these experiments according to the initial objectives.

Lessons Learned from Experience

Effectiveness and Efficiency. The developed implementation of the method presented in section 5.2.1 and the underlying constraint solvers (i.e. Z3 and SICStus) have been shown to be effective and efficient for the intended verification computation. Indeed, the tool chain was always able to conclude about the validity of modal specification over workflow nets of growing size and complexity within the allowed time of 10 minutes (for each constraint system to solve, at least one resolution method was indeed able to assign a verdict, and furthermore within only few seconds in almost all cases). Furthermore, we have observed that memory usage does not seem to be a limiting factor over instances of the considered sizes. Indeed, for the biggest workflow net instance considered, the memory usage for SICStus and Z3 was less than 350MB. finally, we note that, as expected, this extended modal specification method performs efficiently when verifying the conformity of workflow nets with respect to *necessary* and *inadmissible* behaviours thanks to adequate use of over-approximations.

Scalability. On the basis of the results, we can confidently state that the verification method proposed in Chapter 3 is scalable over workflow nets of size up to 500 nodes. Indeed, our experimental results have shown that the proposed verification method has the ability to assign a verdict about the (in)validity of a given modal specification over such workflow nets of growing size and complexity. They also have shown that such verdict is obtained within the admissible time of 10 minutes.

SMT vs CLP. According to these experiments, we can infer that the SMT approach (computed using Z3) generally performs significantly better than the CLP one (computed using SICStus). However, they also highlight that the CLP approach is especially effective and efficient when verifying modal specifications over marked-graph workflow nets. Indeed, we observed that the CLP approach is less efficient than the SMT approach when the number of choice points increases as shown by the results over State-Machine workflow nets. It stems from the labelling done after constraints propagation by CLP solvers which means that an exponential number of backtracking steps may occur with respect to the number of pending choice points.

5.4.3/ Experimental Evaluation of Reduction Methods

In Section 4.3 page 79 two reduction methods, one based on the hierarchical representation of workflow nets and one based on reductions rules, are presented as pre-processing steps towards the verification of extended modal specifications (Section 3.1.1 page 44). These reduction methods aim at reducing the size of the analysed workflow nets in order to reduce the complexity of the intended analysis.

This section presents experimental results supporting the benefits provided by these reduction methods to the workflow net verification of behavioural properties such as generalised soundness and correctness with respect to extended modal specifications.

To this end, we first present the objectives of the proposed experimental evaluation. Second, we define the experimental protocol. Finally, the experimental results obtained according to the defined experimental protocol are presented and discussed.

5.4.3.1/ Objectives

The objectives of this experimental evaluation are to experimentally assess the *effectiveness*, *efficiency* and *scalability* of the proposed reduction methods.

In the context of this experimentation, the effectiveness of the proposed reduction methods is measured with respect to their ability to reduce the size (number of places and transitions) of the considered workflow nets. Indeed, given a workflow net of size $OriginalSize$, the effectiveness of the proposed reduction methods is given by the ratio $ReducedSize/OriginalSize$, where $ReducedSize$ is the size of the workflow net obtained after reduction. Furthermore, the efficiency of the proposed methods is evaluated with respect to the time they required to reduce the considered workflow nets. It follows that the proposed methods are scalable over a considered set of workflow nets of growing size if they are able to effectively and efficiently reduce them.

5.4.3.2/ Experimental Protocol

On the basis of the previously introduced objectives, the following experimental protocol has been designed.

In a first step, this experimental protocol considers the hierarchical workflow net modelling of the issue tracking system presented in Section 5.1.1 page 90. This modelling is expressed by an underlying workflow net composed of 165 places and 204 transitions (369 nodes). For this workflow net, the experimental protocol consists in applying the proposed reduction methods in order to determine its generalised soundness. Additionally, in order to evaluate the effectiveness and efficiency of the proposed reductions methods when preserving the (in)validity of extended modal specification, it also consists in applying the proposed reduction methods as a pre-precessing step towards the verification of the extended modal specifications listed in Table 5.1 page 91.

In a second step, in order to assess more broadly the effectiveness and efficiency of the reduction procedure based on reduction rules introduced in Section 4.2, this experimental protocol consider the benchmark suite of 1976 industrial workflow nets previously studied in [Mendling et al., 2006, van Dongen et al., 2007, Mendling et al., 2008, Fahland et al., 2009] and more recently [Esparza et al., 2016, Favre et al., 2016] where others reduction procedure also based on reduction rules have been applied. This benchmark suite is composed of two main benchmarks. The first one, denoted *IBM-bpm*, is a collection of 1386 free choice workflow nets that have been derived from industrial business process models provided by IBM®. These 1386 models are organized into five libraries (A, B1, B2, B3, and C). They have been translated into Petri nets from IBM Web-Sphere Business Modeler[5]'s language (i.e. a language similar to UML activity diagrams [Dumas et al., 2001]) according to [Fahland, 2008]. The resulting Petri nets often have multiple sink places and have therefore been completed according to [Kiepuszewski et al., 2003] in order to obtain workflow nets. We notably point out that four of the largest workflow nets of this data set are included in the benchmark used by the 2016 Edition of the Model Checking Contest [Kordon et al., 2016]. More information about this data set can be found in the reference paper [Fahland et al., 2009]. The second one, denoted *SAP-ref*, is a collection of 590 workflow nets that have been derived from SAP®'s ERP Software[6] reference models. These 590 industrial workflows have been translated into workflow nets from their original EPCs models [Lohmann et al., 2009].More information about this data set can be found in the reference papers [Mendling et al., 2006, van Dongen et al., 2007, Mendling et al., 2008].

In a third step, in order to further demonstrate the efficiency and scalability of the proposed reduction method based on reduction rules, two different and complementary sets of workflow nets

[5] www.ibm.com/software/products/en/modeler-basic

[6] http://go.sap.com/product/enterprise-management/erp.html

generated using the tool chain presented in Section 5.4.1 are considered. The first set (*gen1*) is composed of 48 generalised sound workflow nets with size ranging from 371 nodes (165 transitions) to 17815 nodes (9233 transitions). The second set (*gen2*) is composed of 59 generalised sound workflow nets with size ranging from 1836 nodes (1000 transitions) to 147640 nodes (75782 transitions). Note that, since the synthesis rules used to build these workflow nets are the inverse of the reduction rules that are used to reduce them, these workflow nets are completely reducible. Therefore the following results are not dedicated to evaluate the effectiveness of the procedure but the efficiency with which it can be applied to workflow nets of growing size.

5.4.3.3/ Results and Feedback from Experiments

This section presents the experimental results obtained using the dedicated reduction tool described in Sections 5.2.2 by applying the experimental protocol introduced in the previous section. All experimental results have been obtained on a computer featuring an Intel(R) Xeon(R) CPU X5650 @ 2.67GHz. Note that only a single core has been used by the implemented tools.

In a first set, regarding the issue tracking system presented in Section 5.1.1 page 90, its underlying workflow net has been completely reduced to the atomic workflow net in 43 ms. This enabled analysts to conclude that this workflow net is generalised sound in such a short time. Furthermore, let us consider the set of requirements to be verified over this modelling given in Table 5.1 page 91. For these requirements, expressed by extended modal specifications in Table 5.8 page 105, Figure 5.4.3.3 presents data about the reduced workflow nets obtained after applications of the reduction methods while preserving the (in)validity of extended modal specification formulae interpreted as *may*-formula or *must*-formula.

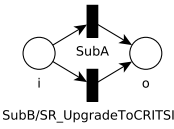
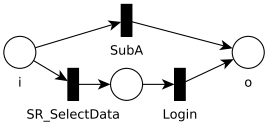
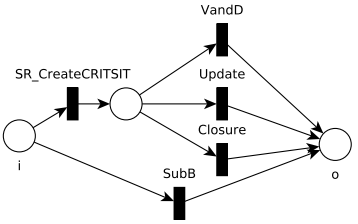
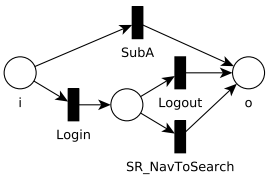
#Property	time(ms)	$\|P\|$	$\|T\|$	$r - factor$	Reduced workflow net
p_1 & p_2	1284	2	2	98.9%	
p_2	1228	3	3	98.3%	
p_3 & p_4	1203	3	5	97.8%	
p_6	1093	3	4	98.1%	

Table 5.15: Reduction Results over the Issue Tracking System

For every property (#Property), the processing time, the number of places ($|P|$) and the number of transitions ($|T|$) of the obtained workflow net as well as its graphical representation are given. This table also reports, for every property, the reduction factor ($r - factor$) obtained. This reduction factor expresses, in percentage, the size reduction with respect to the original workflow net accomplished by the reduction methods used.

These results show that the presented reduction methods are effective and can greatly reduce the size of workflow nets analysed with respect to a given extended modal specification. Indeed, for the considered extended modal specifications, an average reduction factor of 98.3% has been achieved. The obtained reduced nets are very simple (i.e. are composed of only a few transitions and places). Note that these results are not surprising given the fact that the considered workflow net can be completely reduced to prove its generalised soundness. Furthermore, the needed processing time is reasonably low (i.e. about 40 ms per formula). It follows that these reduction methods are effective and efficient over this study case and can greatly enhance its analysis.

In a second step, let us consider the experimental results obtained over the 1976 industrial workflow nets of the data sets *IBM-bpm* and *SAP-ref*. Figures 5.25 and 5.26 respectively present the reduction factors and the sizes of workflow nets of the *IBM-bpm* and *SAP-ref* benchmarks.

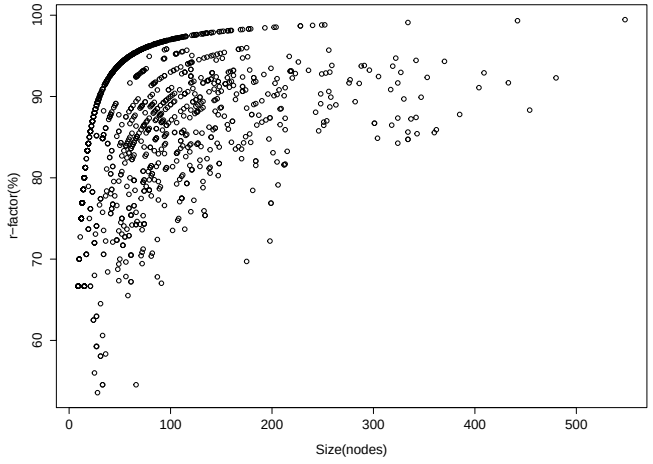


Figure 5.25: Reduction factors of workflow nets of *IBM-bpm*

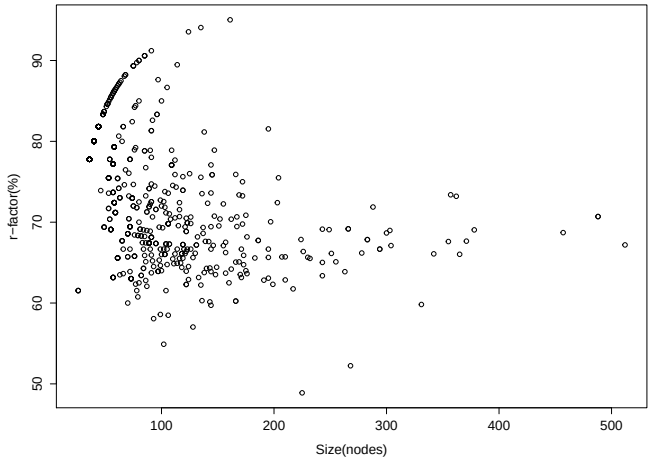


Figure 5.26: Reduction factors of workflow nets of *SAP-ref*

The complete data-sets as well as the obtained experimental results are accessible at https://dx.doi.org/10.6084/m9.figshare.3573756.v5. Table 5.4.3.3 summarizes the results of our experiments for the *IBM-bpm* and *SAP-ref* benchmarks.

Benchmark	Status	#workflow nets	#Nodes		r-factor(%)		time(ms)	
			avg.	max.	avg.	max.	avg.	max.
IBM-bpm(libA)	Sound	152	61.4	193	93.5	98.4	3.2	44
	unknown	130	99.7	277	86.8	95.1	9.6	51
IBM-bpm(libB1)	Sound	107	38.6	228	86.7	98.7	3.8	67
	unknown	181	98.6	360	82	95.8	7.1	54
IBM-bpm(libB2)	Sound	161	38.9	334	85	99.1	7.3	380
	unknown	202	110.3	404	83	95.8	8.5	60
IBM-bpm(libB3)	Sound	207	47.5	252	87.8	98.8	3.9	56
	unknown	214	125.7	454	85.2	96	8.7	72
IBM-bpm(libC)	Sound	15	127	548	94.4	99.5	7.1	46
	unknown	17	135.6	480	84.7	94.9	5.1	28
***IBM-bpm*(all)**	Sound	642	49	548	**88.4**	99.5	4.6	380
	unknown	744	110.6	480	**84.1**	96	8.3	72
SAP-ref	Sound	0	–	–	–	–	–	–
	unknown	590	97.7	512	**73**	95	9.9	967

Table 5.16: Results for the *IBM-bpm* and *SAP-ref* benchmarks

We have analysed 1976 industrial workflow nets and determined that 642 of them are generalised sound (i.e. have been completely reduced). In addition, on average each workflow has been reduced by a factor of 82.2%. These experimental results highlight the effectiveness of the proposed reduction procedure. More precisely, with regards to the 1386 free choice workflow nets of *IBM-bpm*, 642 of them were identified as generalised sound. This results (i.e. the detected soundness) are in agreement with the results obtained during previous experiments applying different analysis techniques to the same data [van Dongen et al., 2007, Favre et al., 2016]. We note that all 642 sound workflow nets have been identified through structural reduction by our approach whereas only 464 of them have been identified through structural reduction by Woflan [Verbeek et al., 2000]. We also note that the reduction method introduced in [Esparza et al., 2016] have found only 470 of these sound workflow nets. Furthermore, let us point out that workflow nets of this data set have been on average reduced to 13.9% of their original size (i.e. reduced by a factor of 86.1%). This underlines the greater effectiveness of our method compared with the reduction method of [Esparza et al., 2016] where workflow nets of the same data set have only been reduced to about 23% of their original size (i.e. reduced by a factor of about 77%).

To further emphasize the effectiveness of our reduction method let us focus on the results obtained for workflow nets of *IBM-bpm* that are included in the benchmark used by the 2016 Edition of the Model Checking Contest [Kordon et al., 2016]. For every considered workflow net, Table 5.4.3.3 gives the number of places ($|P|$) and the number of transitions ($|T|$) of the original and reduced workflow nets. It also states the obtained reduction factor ($r - factor$) as well as the processing time needed to compute it.

These workflow nets, which have been reduced by a factors ranging from 88% to 99.4%, clearly illustrate the benefit of our reduction procedure and its ability to significantly reduce the size of free choice workflow nets regardless of their soundness.

Workflow name	$\|P\|$	$\|T\|$	$r - factor$	t(ms)
IBMB2S565S3960 (original)	274	180	88.3%	53
IBMB2S565S3960 (reduced)	28	25		
IBM5964 (original)	264	140	90.0%	29
IBM5964 (reduced)	19	17		
IBM319 (original)	254	179	91.6%	22
IBM319 (reduced)	19	17		
IBM703 (original)	263	285	99.4%	42
IBM703 (reduced)	2	1		

Table 5.17: Reduction results over the workflow nets of the Model Checking Contest

Regarding the 590 workflow nets of *SAP-ref*, only three of them are free choice workflow nets. None of those 590 workflow nets has been found to be generalised sound by our method. However, workflow nets of this data set have been on average reduced to 27% of their original size (i.e. reduced by a factor of 73%). These results further highlight the effectiveness of our reduction over arbitrary workflow nets. Indeed, once more this results show that our reduction method ability exceed the one of [Esparza et al., 2016] where workflow nets of the same data set have been reduced to about 35% of their original size (i.e. reduced by a factor of about 65%).

The results of our experiments for the *IBM-bpm* and *SAP-ref* benchmarks have also illustrated the efficiency of the proposed reduction method. Indeed, workflow nets of these benchmarks whose sizes are ranging from 9 to 548 nodes have been reduced on average in about 8 ms. Such a short analysis time means that this procedure could be automatically applied by integrated development environment to provide feedback as well as useful diagnostic information regarding the generalised soundness of in-development workflow nets.

In a third step, regarding that data-sets *gen1* and *gen2*. Figures 5.27 and 5.28 present the time, given by the y-axis in seconds or minutes, needed to completely reduce the considered workflow nets whose size (i.e. the number of nodes) is given by the x-axis. The experimental results depicted in Figure 5.27 have deliberately been limited by a verification time bounded to 180 seconds, unlike the results depicted in Figure 5.28 that have been computed without time upper bound.

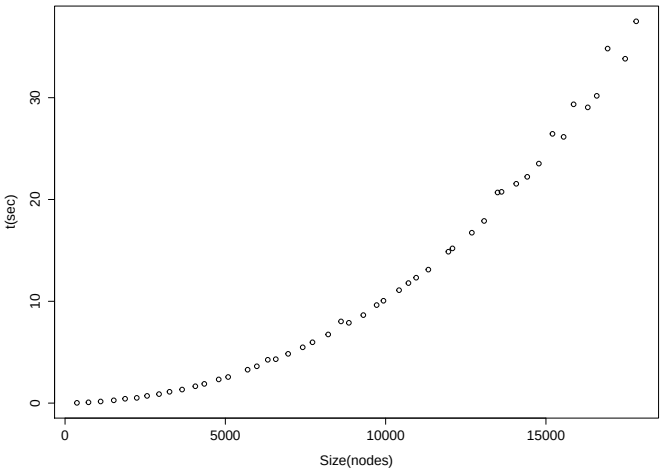


Figure 5.27: Reduction time of workflow nets of the *gen1*

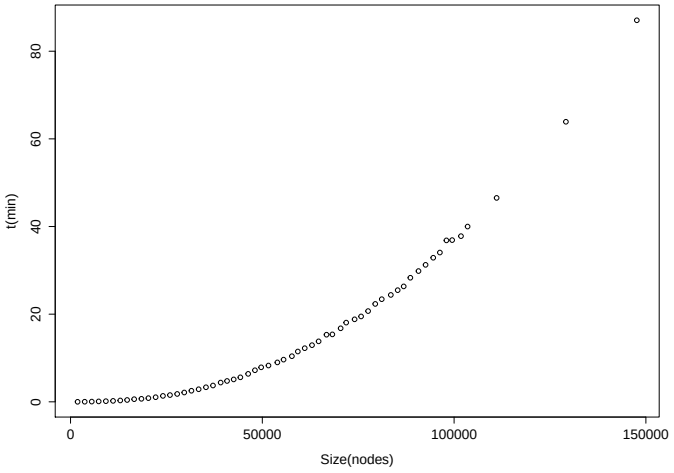


Figure 5.28: Reduction time of workflow nets of the *gen2*

These results allow us to empirically conclude that the proposed reduction procedure can efficiently be applied over large workflow nets (up to 147640 nodes) despite the fact that the application of rules R_1 (defined page 69) and R_2 (defined page 70) is exponential w.r.t. the size of the set $({}^\bullet n)^\bullet$ of the considered place/transition n.

They have also highlighted that the implementation of the proposed method is able to conclude about the generalised soundness of workflow nets of size up to 17815 nodes within the fixed timescale of 180 seconds. In comparison, the approach proposed by Woflan in [Verbeek et al., 2001] is not able to conclude about the 1-soundness of any of these generated workflow nets in the fixed timescale of 180 seconds. Furthermore, the SPIN model-checker is only able to conclude within 180 seconds about the 1-soundness of workflow nets, from our first set of generated workflow nets, of size up to 731 nodes, using the approach proposed in [Yamaguchi et al., 2009]. However, this is not so surprising since both compared approaches rely on state-space exploration and suffer from the state explosion of large-size workflow nets, contrary to the proposed method.

Moreover, as shown in Figure 5.28, the experiments have empirically demonstrated the huge scalability of the proposed method and its implementation within the tool *Hadara-AdSimul-Red*. This tool is indeed able to conclude in less than 90 minutes about the generalised soundness of workflow nets of size at least up to 147640 nodes. This size is clearly far beyond what is realistically produced and manageable by workflows modellers in real-life conditions. Needless to say that their analysis is out of reach of other state exploration approaches (e.g., [Verbeek et al., 2001], [Yamaguchi et al., 2009]).

According to all presented experimental results, it follows that the reduction methods presented in Sections 4.2 page 77 and 4.3 page 79 constitute effective, efficient, scalable and therefore relevant pre-processing steps toward the analysis of behavioural properties such as generalised soundness and correctness with respect to extended modal specifications.

5.5/ SYNTHESIS

This chapter presented convincing experimental results supporting the value of the contributions made to this thesis.

It first presented three study cases (i.e. real-life examples) which highlighted and motivated the use of workflow nets and extended modal specifications. It also described dedicated tool chains implemented to carry out the verification of extended modal specifications according to the approach presented in Section 3.1.3 page 53 as well as the reduction methods of workflow nets presented in Sections 4.2 page 77 and 4.3 page 79.

It then reported experimental results obtained during the analysis of the three study cases using the dedicated modal specification verification tools previously introduced. From these results we concluded that extended modal specifications are adapted to the description of modal behaviour involving several transitions as well as their causalities expressed over workflows modelled as workflow nets but also as coloured workflow nets. Furthermore, the verification of such extended modal specifications was shown to be possible and efficient (less than a few seconds per extended modal formula) over these real-life examples.

This chapter also exposed experimental results regarding the scalability of the proposed approaches over workflow nets of growing size and complexity. To this end, it presented a generation tool able to generate large benchmarks of workflow nets together with their (in)valid extended modal specifications. Based on this generation tool, an experimental protocol and the results it produced were described. These results enabled us to confidently state that the extended modal specification verification method proposed in Section 3.1.3 page 53 is scalable in terms of modal specification and workflow net complexity, as well as regarding their size (up to 500 nodes, at least). It also enabled to conclude that the developed implementation of this verification method and the underlying constraint solvers (i.e. Z3 and SICStus) are effective and efficient for the intended verification computation. More precisely, it showed that it was always able to conclude about the (in)validity of modal specification over workflow nets of growing size and complexity within the allowed time of 10 minutes, and furthermore within only few seconds in almost all cases.

Finally, this chapter also presented experimental results concerning the reduction methods described in Sections 4.2 page 77 and 4.3 page 79. Through one of the considered study cases as well as workflow nets issued from a benchmark suite of 1976 industrial workflow nets previously studied in [van Dongen et al., 2007, Fahland et al., 2009, Favre et al., 2016, Esparza et al., 2016], it demonstrated the size reductions benefits they provide. More precisely, it showed that the verification of extended modal specifications can be carried out over workflow nets whose size is significantly reduced (up to 98.9% smaller than the original workflow nets). Likewise, it also illustrated the effectiveness of such a reduction method to the verification of generalised soundness by providing, for the considered workflow nets, an average size reduction of 82.2%. At last, the efficiency of the developed reduction tool has been demonstrated. It was shown that despite the exponential complexity of some of the reduction rules used, the developed reduction tool was able to reduce workflow nets of size up to 17,815 nodes (resp. 147,640 nodes) in less that 180 seconds (resp. 90 minutes). These observations supported the fact that the reduction methods presented in Sections 4.2 page 77 and 4.3 page 79 constitute relevant, efficient, and scalable pre-processing steps toward the analysis of behavioural properties such as generalised soundness and correctness with respect to extended modal specifications.

III

Conclusion and Future Work

6

Conclusion

Contents

6.1 Verification of Modal specifications 127
6.2 Reduction methods 128
6.3 Experimental Evaluation 129

Nowadays workflow management has become a key factor in the success of companies and organizations around the world. Indeed, workflows, when properly designed, can significantly improve organizational efficiency, responsiveness and profitability by managing the tasks and steps of business processes.

Due to the central role of workflows for day to day operations and long term focus of companies and organizations, the verification of specifications, i.e. of desired properties of workflows, has become mandatory to ensure that such processes are properly designed and reach the expected level of trust and quality.

Within this context, this thesis presented contributions made to the verification of modal specifications of workflow nets based on powerful abstractions and the use of constraint solving.

More precisely, this thesis has addressed the three following research questions:

RQ_1 *How to effectively express modal behaviour of workflow nets involving several transitions and their causalities*?

RQ_2 *How to effectively verify the validity of such modal specifications by leveraging the efficiency of existing mature constraint solvers?*

RQ_3 *Which abstraction mechanisms are appropriate and relevant with respect to verification of such modal specifications?*

The rest of this section briefly summarises the theoretical and practical contributions made to this thesis. Following the organisation of this manuscript, the presentation of these contributions is divided into three subsections.

6.1/ Verification of Modal specifications

To guide engineers in their specification and validation activities, modal specifications have been previously designed to allow specifications of modal behaviour (i.e. admissible or necessary behaviour) of complex systems.

Modal specifications of workflow nets are usually expressed over a single transition of the system under analysis. This is quite limiting as the expression of complex modal behaviour involving several transitions and their causalities is needed in real-life. To address this issue and RQ_1, the research question prompted by it, we defined extended modal specifications, an extension of modal specification that enables the definition of modal behaviour involving several transitions as well as their causalities. Such extended modal specifications have then been further extended to handle workflow nets extensions (e.g., workflow nets with data). To this end, additional constraints on the initial and final states as well as on the data associated to transition firing have been designed and taken into account.

Verifying behavioural properties of workflows such as extended modal specifications is a very complex task which requires exponential computational resources with respect to the size of their modelling by workflow nets. To cope with this complexity, efficient tools are required. Existing mature and efficient constraint solvers offer an unprecedented support to such verification problem.

In order to benefit from it and address RQ_2, a novel constraint-based framework for modelling workflow nets executions has been developed. This modelling framework is based on the definition of several constraint systems either over-approximating or under-approximating the set of correct executions of workflow nets. More precisely, each of these approximations is built by refining the previous one starting from a well-know over-approximation: the state equation. This led to the definition of a constraint system whose solution space under-approximates the set of correct executions of workflow nets. This latter constraint system is used to define segments of execution (i.e. partial execution structurally verifiable). It has been then demonstrated that the concatenation of such segments can be used to model any correct executions of workflow nets.

Furthermore, through the definition of abstract workflow nets – a suited abstract representation notably able to model ordinary, generalised and coloured Petri nets – this framework has been shown to be applicable to workflow nets with data and quantitative performance specification.

Its use for verifying the (in)validity of extended modal specifications of workflow nets (resp. abstract workflow nets) have been presented. Using such a constraint-based modelling for verifying the (in)validity of extended modal specifications of workflow nets (resp. abstract workflow nets) exhibits several advantages. The first advantage of this modelling is that the search is quite focussed from the beginning as we traverse the solution space of the state equation and its refinements rather than the underlying semantic labelled transition systems of workflow nets. Another advantage of such modelling is that only the segments ordering is computed, the transitions ordering within segments is not. It follows that this modelling is particularly well-adapted to the verification of extended modal specification as only the presence or absence of transitions is considered. Moreover, this extended modal specification verification approach requires the use of constraint systems satisfiability checks which can be handled by third party constraint solvers. This allows the proposed verification method to benefit from existing mature and efficient constraint solvers.

6.2/ Reduction methods

The development of large and intricate workflow nets can be a difficult task which requires powerful structuring mechanisms. Unfortunately, most often, the abstraction mechanisms, used by modellers of workflow nets, is not explicitly given beside the clear advantages they bring to their analysis. To cope with this issue and address RQ_3, reduction methods – abstraction methods – that have the ability to reduce the size of workflow nets while strongly preserving properties of interest such as generalised soundness and the (in)validity of a given extended modal specification have been presented. More precisely, six reduction rules preserving generalised soundness and

generalising existing reduction rules have been defined.

Based on the definition of these reduction rules, a generalised soundness semi-decision have been described. We have notably shown that, despite not being complete, this procedure, whenever unable to conclude, can be used as a pre-processing step toward verification of generalised soundness. Indeed, by reducing the size of the analysed workflow nets, this procedure can enhance conventional generalised soundness verification methods. Furthermore, whenever inconclusive, this procedure produces diagnostic information in the form of a simplified and irreducible workflow net.

In addition, two reduction methods preserving the (in)validity of a given extended modal specification has been introduced. The first of these methods – based on the hierarchical representation of workflow nets – aims at abstracting unnecessary layers of detail. The second method is based on the six reduction rules previously presented and further refines them in order to preserve the (in)validity of a given extended modal specification. Both reduction methods were proved to be sound. Thanks to the potential size reduction they provide, these reduction methods have been proposed as pre-processing steps for extended modal specifications verification approaches such as the constraint-based approach previously described.

6.3/ Experimental Evaluation

As a practical contribution to RQ_2 and RQ_3, the new formal approaches described in this thesis have been implemented and extensive experimentations have been carried out in order to validate them.

The results of convincing experimentations carried out over real-life industrial study cases have been introduced and discussed. These results highlight the need and relevance of extended modal specifications to express modal behaviour involving several transitions as well as their causalities. They also illustrate the effectiveness of the proposed modal specification verification approach over workflow nets extended with data and quantitative performance specification.

Furthermore, an empirical evaluation of the effectiveness, efficiency and scalability of the proposed modal specification verification approach over workflow nets of growing size and complexity has been presented. These results show that the extended modal specification verification method proposed in this thesis is effective and scalable in terms of modal specification and workflow net complexity, as well as regarding their size (up to 500 nodes, at least). They also show that the developed implementation of this verification method and the underlying constraint solvers are very efficient for the intended verification computation. More precisely, they point out that the proposed verification approach is able to conclude about the (in)validity of modal specification over workflow nets of growing size and complexity within the allowed time of 10 minutes, and furthermore within only few seconds in almost all cases.

Finally, experimental results demonstrating the benefits provided by the presented reduction methods to the verification of workflow net behavioural properties such as generalised soundness and correctness with respect to modal specifications have been introduced. We have considered one of the study cases as well as workflow nets issued from a benchmark suite of 1976 industrial workflow nets previously studied in [van Dongen et al., 2007, Fahland et al., 2009, Favre et al., 2016, Esparza et al., 2016]. We have shown that the verification of extended modal specifications can be carried out over workflow nets whose size is significantly reduced (up to 98.9% smaller than the original workflow nets). Likewise, we have also illustrated the effectiveness of such a reduction method to the verification of generalised soundness. Indeed, experimental results for 1976

workflow nets derived from industrial business processes show that the nets are reduced to about 17.8% of their original size (i.e. reduced by a factor of 82.2%). At last, the efficiency of the developed reduction tool has been demonstrated. It was shown that despite the exponential complexity of some of the reduction rules used, the developed reduction tool was able to reduce workflow nets of size up to 17,815 nodes (resp. 147,640 nodes) in less that 180 seconds (resp. 90 minutes). Based on these observations we concluded that the presented reduction methods constitute effective, efficient, and scalable pre-processing steps toward the analysis of behavioural properties such as generalised soundness and correctness of workflow nets with respect to extended modal specifications.

7

Future Work

Contents

7.1 Towards Parallelism 131
7.2 Error-pattern 132
7.3 Reconfiguration 133

This section presents three future work directions prompted by the research work and results of this thesis. We first discuss the adaptations that would enable the application of the presented approaches to benefit from parallel computing. Afterwards, we present an extension of the proposed generalised soundness semi-decision procedure which introduces the identification of *error-patterns* during the reduction process in order to provide insight about the invalidity of workflow nets with respect to generalised soundness. Finally, we propose a reconfiguration framework which enables the definition of workflow net reconfigurations preserving behavioural properties such as generalised soundness and validity with respect to extended modal specifications.

7.1/ Towards Parallelism

Parallel computing is a type of computation in which many calculations are carried out simultaneously [Almasi et al., 1988]. Large problems can often be divided into smaller ones, which can then be solved independently at the same time.

Within parallel computing, there exist two main paradigms: shared and distributed memory programming [Kumar et al., 1994]. On the one hand, in shared memory programming, all tasks access the same memory and therefore the same data (e.g., POSIX Threads [Butenhof, 1997]). On the other hand, in distributed memory programming, all memory is local and data sharing is achieved by explicitly transporting data through communication (e.g., MPI [Gropp et al., 1999]).

With regards to the proposed extended modal specification verification approach based on constraint systems, note that the computation workload of the implemented tool is shifted to mature and efficient but sequential solvers. Due to the increasing demand for high performance constraint solving algorithms in the industry, parallel constraint solvers constitute an active research area [Singer, 2006, Hamadi et al., 2013, Hyvärinen et al., 2016]. In a future work, parallel constraint solvers could substitute the sequential solvers used in the considered implementation in order to further improve its efficiency.

With regards to the proposed reduction methods, note that, given a workflow net, several reduction rules may be simultaneously applied to distinct nodes. This means that the application of the presented reduction procedures based on reduction rules could in the future be implemented within a shared memory environment (e.g., actual multi-core computer). Such an implementation

would be very similar to the one we implemented with the difference that it would require synchronisation such as a mutual exclusion mechanism to avoid concurrent modification of the same nodes. Due to the large number of reduction rules applicable to distinct nodes observed over the industrial workflow nets considered in this thesis, we conjecture that such a shared memory parallel implementation of the proposed reduction procedures would significantly improve their overall efficiency. Furthermore, note that, given a hierarchical workflow net, each of its nodes can be reduced independently. It follows that each of these nodes can simultaneously be reduced in a distributed memory environment (e.g., cluster, grid). Depending on the number of hierarchical levels of detail, such a distributed implementation of the proposed reduction methods could also significantly improve their overall efficiency. Finally, note that those shared and distributed approach to the reduction of workflow nets are not mutually exclusive and could be combined in a mixed environment with both shared and distributed memory ability (e.g., accelerators: Intel® Xeon Phi cluster).

7.2/ Error-pattern

We designed and presented reduction rules of workflow nets notably preserving generalised soundness. The idea behind this concept is to study the construction of workflow nets. If a workflow net is generalised sound, chances are that it has been modelled using strict abstraction mechanisms either implicitly or explicitly in the case of refinement development. It follows that by identifying sound abstraction mechanisms used by modellers we may be able to prove the considered property. In the context of generalised soundness this has led to the design of the presented reduction rules as well as their dual synthesis rules corresponding to sound abstraction mechanisms with respect to generalised soundness. These reduction rules, as it has been presented, when iteratively applied, form a semi-decision procedure for generalised soundness. Indeed, as these reduction rules are not complete, no negative results can be inferred.

A natural alternative to this approach consists in identifying workflow patterns contradicting the generalised soundness property. Such patterns are called error-patterns and correspond to defects. Recently, they have notably been used in order to provide diagnostic information for control-flow analysis of free-choice workflow nets [Favre et al., 2016]. We propose to apply a similar concept to the analysis of arbitrary workflow nets with respect to the generalised soundness property. Indeed, if a workflow net is not generalised sound, chances are that at some point in its construction an error-pattern has been introduced. Let us for example consider the following theorem.

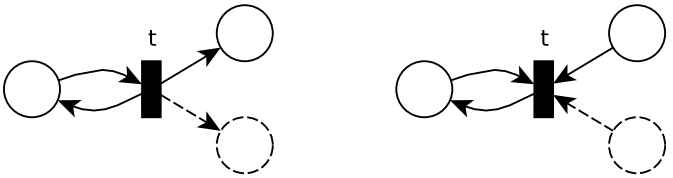
Theorem 36: Example of error-pattern

Let $N = \langle P, T, F \rangle$ be a workflow net. If there exists a transition $t \in T$ such that ${}^\bullet t \neq t^\bullet \wedge ({}^\bullet t \setminus t^\bullet = \emptyset \vee t^\bullet \setminus {}^\bullet t = \emptyset)$ then N is not generalised sound.

Proof. We proceed to demonstrate this theorem by contradiction. Assume that N is a generalised sound workflow net. Suppose there exists a transition $t \in T$ such that ${}^\bullet t \neq t^\bullet \wedge {}^\bullet t \setminus t^\bullet = \emptyset$. We have ${}^\bullet t \subseteq t^\bullet$ and by assumption there exists a marking $M \in M \in \mathcal{R}^N(M_{i(1)})$ such that t is enabled. It follows that in marking M, the transition t can be fired infinitely many times. This would produce an infinite number of tokens in places of $t^\bullet \setminus {}^\bullet t$. The 1-closure of N would not be bounded and this would therefore contradict the assumption. Now, suppose there exists a transition $t \in T$ such that ${}^\bullet t \neq t^\bullet \wedge t^\bullet \setminus {}^\bullet t = \emptyset$. We have $t^\bullet \subseteq {}^\bullet t$ and therefore the firing of t does not produce any new tokens. By assumption, there exist σ_1, σ_2 two executions of N, and M, M' two markings of N such that $M_{i(1)} \xrightarrow{\sigma_1} M \xrightarrow{t} M' \xrightarrow{\sigma_1} M_{o(1)}$. It follows that the concatenation of executions σ_1 and σ_2 leads to a

marking M_f such that $\forall\ p \in {}^\bullet t \setminus t^\bullet,\ M_f(p) = 1$ and $M_f(o) = 1$. Therefore we have $M_f \in \mathcal{R}^N(M_{i(1)})$ and this contradicts the assumption. □

This theorem defines infinitely many error-patterns. Figures 7.1(a) and 7.1(b) illustrate some of them.



(a) (b)

Figure 7.1: Illustration of error-patterns

Our experimental results have pointed out that error-patterns such as the ones described by Theorem 36 often appear during the reduction process. Indeed, we have observed that while many defects are initially hidden by the complexity of the considered workflow net, the reduction process tends to reduce their expressions to identifiable error-patterns. It follows that the search of error-patterns during the reduction process could provide insight with regards to the invalidity of workflow nets with respect to generalised soundness. The synergy of these two complementary approaches would result in an algorithm semi-deciding both the validity and the invalidity of workflow nets with respect to generalised soundness and should greatly enhance reduction based analysis abilities.

7.3/ Reconfiguration

In order to succeed long into the future, a business needs to be flexible and constantly evolves with respect to its industry. Therefore the workflows supporting the business processes also have to evolve in order to be up to date.

In the context of workflow nets, such evolutions of the workflows are modelled through *reconfigurations*. There exist two types of reconfigurations: static and dynamic. Static reconfigurations are traditional updates. Following requests of users or technical imperatives, new versions of workflow nets are developed based on the actual versions before being deployed. Such reconfigurations often take time and might not be sufficiently responsive. To cope with this, dynamic reconfigurations can be used. Based on some pre-conditions (e.g., overload, failure of a task), predefined reconfigurations can be applied dynamically during execution.

A major aspect of reconfigurations is the fact that they must preserve behavioural properties of interest. For example, reconfigurations of workflow nets should at least preserve generalised soundness. Further, they should also preserve one or more extended modal specifications.

In Section 4.3 page 79, six reduction rules strongly preserving generalised soundness as well as the (in)validity of a given extended modal specification formula interpreted as either a *may*-formula or a *must*-formula have been introduced. Note that the inverse of these reduction rules are six synthesis rules, which also preserve generalised soundness as well as the (in)validity of a given extended modal specification formula interpreted as either a *may*-formula or a *must*-formula.

We propose to model the changes made to workflow nets by reconfigurations as a sequences of synthesis and reduction rules such as the ones defined in this thesis. Doing so would have the advantage to guarantee the validity of behavioural properties without having resort to the in-depth analysis of the behavioural properties of interest. A sequence of synthesis and reduction rules would model a reconfiguration which preserves not only generalised soundness but also the (in)validity of a given extended modal specification formula interpreted as either a *may*-formula or a *must*-formula if necessary.

Note that, in the case of static reconfigurations, the execution of the workflows might be halted to perform reconfigurations. Further, dynamic reconfigurations might not be instantaneous and require a significant downtime. Another advantage of the proposed reconfiguration approach is that it models reconfigurations as sequences of atomic workflow transformations (i.e. synthesis and reduction rules) which can be sequentially applied. Therefore, modifications of the workflow nets are local and fast which should decrease the overall impact of reconfigurations and ensures quality of services.

Bibliography

[Allweyer, 2016] Allweyer, T. (2016). **BPMN 2.0: introduction to the standard for business process modeling**. BoD–Books on Demand.

[Almasi et al., 1988] Almasi, G. S., et Gottlieb, A. (1988). **Highly parallel computing**.

[Ashton et al., 1997] Ashton, T. S., et others (1997). **The industrial revolution 1760-1830**. *OUP Catalogue*.

[Barkaoui et al., 2007] Barkaoui, K., Ben Ayed, R., et Sbai, Z. (2007). **Workflow soundness verification based on structure theory of Petri nets**. *Journal of Computing and Information Sciences*, 5(1):51–61.

[Barrett et al., 2011] Barrett, C., Conway, C. L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., et Tinelli, C. (2011). **Cvc4**. In *Computer aided verification*, pages 171–177. Springer.

[Barrett et al., 2005] Barrett, C., De Moura, L., et Stump, A. (2005). **Smt-comp: Satisfiability modulo theories competition**. In *International Conference on Computer Aided Verification*, pages 20–23. Springer.

[Bayardo et al., 1996] Bayardo, R. J., et Miranker, D. P. (1996). **A complexity analysis of space-bounded learning algorithms for the constraint satisfaction problem**. In *Proceedings Of The National Conference On Artificial Intelligence*, pages 298–304. Citeseer.

[Berthelot, 1987] Berthelot, G. (1987). **Transformations and decompositions of nets**. In *Petri Nets: Central models and their properties*, pages 359–376. Springer.

[Berthomieu* et al., 2004] Berthomieu*, B., Ribet, P.-O., et Vernadat, F. (2004). **The tool tina–construction of abstract state spaces for petri nets and time petri nets**. *International Journal of Production Research*, 42(14):2741–2756.

[Biere et al., 2009] Biere, A., Heule, M., et van Maaren, H. (2009). **Handbook of satisfiability**, volume 185. ios press.

[Bonet et al., 2007] Bonet, P., Lladó, C. M., Puijaner, R., et Knottenbelt, W. J. (2007). **PIPE v2.5: A Petri net tool for performance modelling**. In *Proc. of the 23rd Latin American Conference on Informatics (CLEI'07)*, San Jose, Costa Rica.

[Bordeaux et al., 2006] Bordeaux, L., Hamadi, Y., et Zhang, L. (2006). **Propositional satisfiability and constraint programming: A comparative survey**. *ACM Computing Surveys (CSUR)*, 38(4):12.

[Bourdeaud'Huy et al., 2008] Bourdeaud'Huy, T., Yim, P., et Hanafi, S. (2008). **Incremental integer linear programming models for Petri nets reachability problems**. INTECH Open Access Publisher.

[Bride et al., 2014] Bride, H., Kouchnarenko, O., et Peureux, F. (2014). **Verifying modal workflow specifications using constraint solving**. In *Int. Conf. on Integrated Formal Methods (IFM'14)*, volume 8739 of *LNCS*, pages 171–186, Bertinoro, Italy. Springer.

[Bride et al., 2015] Bride, H., Kouchnarenko, O., et Peureux, F. (2015). **Constraint solving for verifying modal specifications of workflow nets with data**. In *Proceedings of 10th International Ershov Informatics Conference - Perspectives of System Informatics (PSI'15)*, volume 9609 of *LNCS*, pages 75–90, Kazan, Russia. Springer.

[Bride et al., 2016a] Bride, H., Kouchnarenko, O., Peureux, F., et Voiron, G. (2016a). **Comparaison des approches SMT et CSP appliquées à la vérification de réseaux workflows**. In *Actes des 15èmes journées sur les Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL'16)*, pages 11–12, Besançon, France.

[Bride et al., 2016b] Bride, H., Kouchnarenko, O., Peureux, F., et Voiron, G. (2016b). **Workflow nets verification: SMT or CLP?** In *Proceedings of the 21st International Workshop on Formal Methods for Industrial Critical Systems and Automated Verification of Critical Systems (FMICS-AVoCS'16)*, volume 9933 of *LNCS*, pages 1–17, Pisa, Italy. Springer.

[Burroni, 1993] Burroni, A. (1993). **Higher-dimensional word problems with applications to equational logic**. *Theoretical computer science*, 115(1):43–62.

[Butenhof, 1997] Butenhof, D. R. (1997). **Programming with POSIX threads**. Addison-Wesley Professional.

[Carlsson et al., 2012] Carlsson, M., et others (2012). **SICStus Prolog user's manual (Release 4.2.3)**. Swedish Institute of Computer Science, Kista, Sweden.

[Carlsson et al., 1988] Carlsson, M., Widen, J., Andersson, J., Andersson, S., Boortz, K., Nilsson, H., et Sjöland, T. (1988). **SICStus Prolog user's manual**, volume 3. Swedish Institute of Computer Science Kista, Sweden.

[Clarke et al., 2000] Clarke, E., Grumberg, O., Jha, S., Lu, Y., et Veith, H. (2000). **Counterexample-guided abstraction refinement**. In *Computer aided verification*, pages 154–169. Springer.

[Clarke et al., 1981] Clarke, E. M., et Emerson, E. A. (1981). **Design and synthesis of synchronization skeletons using branching time temporal logic**. Springer.

[Clarke et al., 1986] Clarke, E. M., Emerson, E. A., et Sistla, A. P. (1986). **Automatic verification of finite-state concurrent systems using temporal logic specifications**. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263.

[Clarke et al., 1999] Clarke, E. M., Grumberg, O., et Peled, D. (1999). **Model checking**. MIT press.

[Clarke et al., 1996] Clarke, E. M., et Wing, J. M. (1996). **Formal methods: State of the art and future directions**. *ACM Computing Surveys (CSUR)*, 28(4):626–643.

[Coalition, 1996] Coalition, W. M. (1996). **Terminology & glossary**. *WFMC Document WFMCTC-1011, Workflow Management Coalition, Avenue Marcel Thiry*, 204:1200.

[De Giacomo et al., 2013] De Giacomo, G., et Vardi, M. Y. (2013). **Linear temporal logic and linear dynamic logic on finite traces**. In *IJCAI'13 Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 854–860. Association for Computing Machinery.

[De Moura et al., 2008] De Moura, L., et Bjørner, N. (2008). **Z3: An efficient smt solver**. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer.

[De Moura et al., 2011] De Moura, L., et Bjørner, N. (2011). **Satisfiability modulo theories: introduction and applications**. *Communications of the ACM*, 54(9):69–77.

[Dechter et al., 2002] Dechter, R., et Frost, D. (2002). **Backjump-based backtracking for constraint satisfaction problems**. *Artificial Intelligence*, 136(2):147–188.

[Desel, 1998] Desel, J. (1998). **Basic linear algebraic techniques for place/transition nets**. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 257–308. Springer.

[Desel et al., 2005] Desel, J., et Esparza, J. (2005). **Free choice Petri nets**, volume 40. Cambridge university press.

[DeVore et al., 1987] DeVore, I., et Tooby, J. (1987). **The reconstruction of hominid behavioral evolution through strategic modeling**. *The Evolution of Human Behavior: Primate Models, edited by WG Kinzey*, pages 183–237.

[Dittrich, 1989] Dittrich, G. (1989). **Specification with nets**. In *Computer Aided Systems Theory—EUROCAST'89*, pages 111–124. Springer.

[Dumas et al., 2001] Dumas, M., et Ter Hofstede, A. H. (2001). **Uml activity diagrams as a workflow specification language**. In *International Conference on the Unified Modeling Language*, pages 76–90. Springer.

[Elhog-Benzina et al., 2012] Elhog-Benzina, D., Haddad, S., et Hennicker, R. (2012). **Refinement and asynchronous composition of modal petri nets**. In *Transactions on Petri Nets and Other Models of Concurrency V*, volume 6900 of *LNCS*, pages 96–120. Springer.

[Esparza, 1998] Esparza, J. (1998). **Decidability and complexity of Petri net problems — an introduction**. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 374–428. Springer.

[Esparza et al., 2016] Esparza, J., et Hoffmann, P. (2016). **Reduction rules for colored workflow nets**. In *International Conference on Fundamental Approaches to Software Engineering*, pages 342–358. Springer.

[Esparza et al., 2014] Esparza, J., Ledesma-Garza, R., Majumdar, R., Meyer, P., et Niksic, F. (2014). **An smt-based approach to coverability analysis**. In *International Conference on Computer Aided Verification*, pages 603–619. Springer.

[Esparza et al., 2000] Esparza, J., et Melzer, S. (2000). **Verification of safety properties using integer programming: Beyond the state equation**. *Formal Methods in System Design*, 16(2):159–189.

[Esparza et al., 2015] Esparza, J., et Meyer, P. J. (2015). **An smt-based approach to fair termination analysis**. In *Proceedings of the 15th Conference on Formal Methods in Computer-Aided Design*, pages 49–56. FMCAD Inc.

[Fahland, 2008] Fahland, D. (2008). **Translating uml2 activity diagrams to petri nets**.

[Fahland et al., 2009] Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., et Wolf, K. (2009). **Instantaneous soundness checking of industrial business process models**. In *International Conference on Business Process Management*, pages 278–293. Springer.

[Favre et al., 2016] Favre, C., Völzer, H., et Müller, P. (2016). **Diagnostic information for control-flow analysis of workflow graphs (aka free-choice workflow nets)**. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 463–479. Springer.

[Fehling, 1991] Fehling, R. (1991). **A concept of hierarchical petri nets with building blocks**. In *Advances in Petri Nets 1993*, pages 148–168. Springer.

[Fischer, 2003] Fischer, L. (2003). **Workflow handbook 2003**. Future Strategies Inc.

[Frost et al., 1995] Frost, D., Dechter, R., et others (1995). **Look-ahead value ordering for constraint satisfaction problems**. In *IJCAI (1)*, pages 572–578. Citeseer.

[Gabbay et al., 1980] Gabbay, D., Pnueli, A., Shelah, S., et Stavi, J. (1980). **On the temporal analysis of fairness**. In *Proceedings of the 7th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 163–173. ACM.

[Goldman et al., 2012] Goldman, A., et Ngoko, Y. (2012). **On graph reduction for qos prediction of very large web service compositions**. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pages 258–265. IEEE.

[Gropp et al., 1999] Gropp, W., Lusk, E., et Skjellum, A. (1999). **Using MPI: portable parallel programming with the message-passing interface**, volume 1. MIT press.

[Gulwani et al., 2008] Gulwani, S., Srivastava, S., et Venkatesan, R. (2008). **Program analysis as constraint solving**. *ACM SIGPLAN Notices*, 43(6):281–292.

[Hamadi et al., 2013] Hamadi, Y., et Wintersteiger, C. (2013). **Seven challenges in parallel sat solving**. *AI Magazine*, 34(2):99.

[Hayden et al., 1968] Hayden, S., Zermelo, E., Fraenkel, A. A., et Kennison, J. F. (1968). **Zermelo-Fraenkel set theory**. CE Merrill.

[Hichami et al., 2014] Hichami, O. E., Al Achhab, M., Berrada, I., Oucheikh, R., et El Mohajir, B. E. (2014). **An approach of optimisation and formal verification of workflow petri nets.** *Journal of Theoretical & Applied Information Technology*, 61(3).

[Hillah et al., 2010] Hillah, L.-M., Kordon, F., Petrucci, L., et Trèves, N. (2010). **Pnml framework: An extendable reference implementation of the petri net markup language**. In *Applications and Theory of Petri Nets*, pages 318–327. Springer.

[Hyvärinen et al., 2016] Hyvärinen, A. E., Marescotti, M., Alt, L., et Sharygina, N. (2016). **Opensmt2: An smt solver for multi-core and cloud computing**. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 547–553. Springer.

[Jech, 2013] Jech, T. (2013). **Set theory**. Springer Science & Business Media.

[Jensen, 1987] Jensen, K. (1987). **Coloured Petri nets**. In *Petri Nets: Central Models and Their Properties*, volume 254 of *LNCS*, pages 248–299. Springer.

[Karp et al., 1969] Karp, R. M., et Miller, R. E. (1969). **Parallel program schemata**. *Journal of Computer and system Sciences*, 3:147–195.

[Kiepuszewski et al., 2003] Kiepuszewski, B., ter Hofstede, A. H., et van der Aalst, W. M. (2003). **Fundamentals of control flow in workflows**. *Acta Informatica*, 39(3):143–209.

[Kindler, 2004] Kindler, E. (2004). **On the semantics of epcs: A framework for resolving the vicious circle**. In *International Conference on Business Process Management*, pages 82–97. Springer.

[Kleine et al., 2010] Kleine, M., et Göthel, T. (2010). **Specification, verification and implementation of business processes using CSP**. In *TASE*, pages 145–154. IEEE Computer Society.

[Koehler et al., 2014] Koehler, J., Moser, S. D., Vanhatalo, J. H., et Voelzer, H. (2014). **System and method for hierarchically decomposing process model**. US Patent 8,786,602.

[Kordon et al., 2016] Kordon, F., Garavel, H., Hillah, L. M., Hulin-Hubard, F., Chiardo, G., Hamez, A., Jezequel, L., Miner, A., Meijer, J., Paviot-Adet, E., Racordon, D., Rodriguez, C., Rohr, C., Srba, J., Thierry-Mieg, Y., Trịnh, G., et Wolf, K. (2016). **Models of the 2016 Edition of the Model Checking Contest**. http://mcc.lip6.fr/models.php.

[Kosaraju, 1982] Kosaraju, S. R. (1982). **Decidability of reachability in vector addition systems (preliminary version)**. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 267–281. ACM.

[Kumar, 1992] Kumar, V. (1992). **Algorithms for constraint-satisfaction problems: A survey**. *AI magazine*, 13(1):32.

[Kumar et al., 1994] Kumar, V., Grama, A., Gupta, A., et Karypis, G. (1994). **Introduction to parallel computing: design and analysis of algorithms**, volume 400. Benjamin/Cummings Redwood City, CA.

[Larsen, 1989] Larsen, K. G. (1989). **Modal Specifications**. In *Proc. of the Int. Workshop on Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, pages 232–246, Grenoble, France. Springer-Verlag.

[Lawrence, 1997] Lawrence, P. (1997). **Workflow handbook 1997**. John Wiley & Sons, Inc.

[Lee-Kwang et al., 1987] Lee-Kwang, H., Favrel, J., et Baptiste, P. (1987). **Generalized petri net reduction method**. *IEEE transactions on systems, man, and cybernetics*, 17(2):297–303.

[Leroux, 2011] Leroux, J. (2011). **Vector addition system reachability problem: a short self-contained proof**. In *Language and Automata Theory and Applications*, pages 41–64. Springer.

[Li et al., 2009] Li, Z. W., et Zhou, M. C. (2009). **Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach**. Springer Publishing Company, Incorporated, 1st edition. ISBN 184882243X.

[Lin et al., 2002] Lin, H., Zhao, Z., Li, H., et Chen, Z. (2002). **A novel graph reduction algorithm to identify structural conflicts**. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pages 10–pp. IEEE.

[Lipton, 1976] Lipton, R. (1976). **The reachability problem requires exponential space**. Research report (Yale University. Department of Computer Science). Department of Computer Science, Yale University.

[Lohmann et al., 2009] Lohmann, N., Verbeek, E., et Dijkman, R. (2009). **Petri net transformations for business processes–a survey**. In *Transactions on petri nets and other models of concurrency II*, pages 46–63. Springer.

[Marechal et al., 2013] Marechal, A., et Buchs, D. (2013). **Unifying the semantics of modular extensions of petri nets**. In *International Conference on Applications and Theory of Petri Nets and Concurrency*, pages 349–368. Springer.

[Marechal et al., 2015] Marechal, A., et Buchs, D. (2015). **Generalizing the compositions of petri nets modules**. *Fundamenta Informaticae*, 137(1):87–116.

[Melzer et al., 1996] Melzer, S., et Esparza, J. (1996). **Checking system properties via integer programming**. In *Proc. of the 6th Eur. Symp. on Programming Languages and Systems (ESOP'96)*, volume 1058 of *LNCS*, pages 250–264, Linköping, Sweden. Springer.

[Mendling et al., 2006] Mendling, J., Moser, M., Neumann, G., Verbeek, H., Van Dongen, B. F., et van der Aalst, W. M. (2006). **Faulty epcs in the sap reference model**. In *International Conference on Business Process Management*, pages 451–457. Springer.

[Mendling et al., 2008] Mendling, J., Verbeek, H., van Dongen, B. F., van der Aalst, W. M., et Neumann, G. (2008). **Detection and prediction of errors in epcs of the sap reference model**. *Data & Knowledge Engineering*, 64(1):312–329.

[Monakova et al., 2009] Monakova, G., Kopp, O., Leymann, F., Moser, S., et Schäfers, K. (2009). **Verifying business rules using an SMT solver for BPEL processes**. In *BPSC*, volume 147 of *LNI*, pages 81–94. GI.

[Murata, 1989] Murata, T. (1989). **Petri nets: Properties, analysis and applications**. *Proc. of the IEEE*, 77(4):541–580.

[Namjoshi et al., 2000] Namjoshi, K. S., et Kurshan, R. P. (2000). **Syntactic program transformations for automatic abstraction**. In *12th Int. Conf. on Computer Aided Verification (CAV'00)*, volume 1855 of *LNCS*, pages 435–449, Chicago, IL, USA. Springer.

[Nieuwenhuis et al., 2006] Nieuwenhuis, R., Oliveras, A., et Tinelli, C. (2006). **Solving sat and sat modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll (t)**. *Journal of the ACM (JACM)*, 53(6):937–977.

[Oliveras, 2014] Oliveras, A. (2014). **Survey of satisfiability modulo theories (smt)**. In *Banff International Research Station for Mathematical Innovation and Discovery (BIRS) Workshop Lecture Videos*. Banff International Research Station for Mathematical Innovation and Discovery.

[Padberg, 1999] Padberg, J. (1999). **Abstract petri nets as a uniform approach to high-level petri nets**. In Fiadeiro, J., editor, *Recent Trends in Algebraic Development Techniques*, volume 1589 of *Lecture Notes in Computer Science*, pages 241–260. Springer Berlin Heidelberg.

[Petri, 1962] Petri, C. A. (1962). **Kommunikation mit Automaten**. PhD thesis, Darmstadt University of Technology, Germany.

[Ping et al., 2004] Ping, L., Hao, H., et Jian, L. (2004). **On 1-soundness and soundness of workflow nets**. In *Proceedings of the Third Workshop on Modelling of Objects, Components, and Agents Aarhus, Denmark*, pages 21–36.

[Pólrola et al., 2014] Pólrola, A., Cybula, P., et Meski, A. (2014). **Smt-based reachability checking for bounded time Petri nets**. *Fundam. Inform.*, 135(4):467–482.

[Polyvyanyy et al., 2011] Polyvyanyy, A., Weidlich, M., et Weske, M. (2011). **Connectivity of workflow nets: the foundations of stepwise verification**. *Acta informatica*, 48(4):213–242.

[Prasad et al., 2005] Prasad, M. R., Biere, A., et Gupta, A. (2005). **A survey of recent advances in sat-based formal verification**. *International Journal on Software Tools for Technology Transfer*, 7(2):156–173.

[Rabbi et al., 2013] Rabbi, F., Wang, H., MacCaull, W., et Rutle, A. (2013). **A model slicing method for workflow verification**. *Electronic Notes in Theoretical Computer Science*, 295:79–93.

[Raedts et al., 2007] Raedts, I., Petkovic, M., Usenko, Y. S., van der Werf, J. M. E., Groote, J. F., et Somers, L. J. (2007). **Transformation of bpmn models for behaviour analysis.** In *MSVVEIS*, pages 126–137.

[Rakow, 2008] Rakow, A. (2008). **Slicing petri nets with an application to workflow verification**. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 436–447. Springer.

[Ramchandani, 1974] Ramchandani, C. (1974). **Analysis of asynchronous concurrent systems by timed petri nets**.

[Rybalchenko, 2010] Rybalchenko, A. (2010). **Constraint solving for program verification: Theory and practice by example**. In *International Conference on Computer Aided Verification*, pages 57–71. Springer.

[Sadiq et al., 2000] Sadiq, W., et Orlowska, M. E. (2000). **Analyzing process models using graph reduction techniques**. *Information systems*, 25(2):117–134.

[Schäl, 1996] Schäl, T. (1996). **Workflow management for process organisations, volume 1096 of**. *Lecture Notes in Computer Science*.

[Scheer et al., 2005] Scheer, A.-W., Thomas, O., et Adam, O. (2005). **Process modeling using event-driven process chains**. *Process-Aware Information Systems*, pages 119–146.

[Schmidt, 2000] Schmidt, K. (2000). **Lola a low level analyser**. In *International Conference on Application and Theory of Petri Nets*, pages 465–474. Springer.

[Schmidt, 2001] Schmidt, K. (2001). **Narrowing petri net state spaces using the state equation**. *Fundamenta Informaticae*, 47(3-4):325–335.

[Singer, 2006] Singer, D. (2006). **Parallel resolution of the satisfiability problem: A survey**. *Parallel combinatorial optimization*, pages 123–148.

[Sloan et al., 1996] Sloan, R. H., et Buy, U. (1996). **Reduction rules for time petri nets**. *Acta Informatica*, 33(7):687–706.

[Soliman, 2008] Soliman, S. (2008). **Finding minimal P/T-invariants as a CSP**. In *Proc. of the 4th Workshop on Constraint Based Methods for Bioinformatics (WCB'08)*.

[Stuckey et al., 2014] Stuckey, P. J., Feydy, T., Schutt, A., Tack, G., et Fischer, J. (2014). **The minizinc challenge 2008–2013**. *AI Magazine*, 35(2):55–60.

[Suzuki et al., 1983] Suzuki, I., et Murata, T. (1983). **A method for stepwise refinement and abstraction of petri nets**. *Journal of computer and system sciences*, 27(1):51–76.

[Ţiplea et al., 2005] Ţiplea, F. L., et Marinescu, D. C. (2005). **Structural soundness of workflow nets is decidable**. *Information Processing Letters*, 96(2):54–58.

[Tsang, 1993] Tsang, E. (1993). **Foundation of constraint satisfaction**. Academic Press.

[Valmari, 1998] Valmari, A. (1998). **The state explosion problem**. In *Lectures on Petri nets I: Basic models*, pages 429–528. Springer.

[Van Der Aalst et al., 2004] Van Der Aalst, W., et Van Hee, K. M. (2004). **Workflow management: models, methods, and systems**. MIT press.

[Van der Aalst, 1997] Van der Aalst, W. M. (1997). **Verification of workflow nets**. In *Application and Theory of Petri Nets 1997*, pages 407–426. Springer.

[van der Aalst, 1998] van der Aalst, W. M. (1998). **The Application of Petri Nets to Workflow Management**. *Journal of Circuits, Systems, and Computers*, 8(1):21–66.

[van der Aalst et al., 2000] van der Aalst, W. M., Barros, A. P., ter Hofstede, A. H., et Kiepuszewski, B. (2000). **Advanced workflow patterns**. In *International Conference on Cooperative Information Systems*, pages 18–29. Springer.

[van der Aalst et al., 2011] van der Aalst, W. M., van Hee, K. M., ter Hofstede, A. H., Sidorova, N., Verbeek, H., Voorhoeve, M., et Wynn, M. T. (2011). **Soundness of workflow nets: classification, decidability, and analysis**. *Journal of Formal Aspects of Computing*, 23(3):333–363.

[van Dongen et al., 2007] van Dongen, B. F., Jansen-Vullers, M. H., Verbeek, H., et van der Aalst, W. M. (2007). **Verification of the sap reference models using epc reduction, state-space analysis, and invariants**. *Computers in Industry*, 58(6):578–601.

[van Dongen et al., 2005] van Dongen, B. F., Van der Aalst, W. M., et Verbeek, H. M. (2005). **Verification of epcs: Using reduction rules and petri nets**. In *International Conference on Advanced Information Systems Engineering*, pages 372–386. Springer.

[van Hee et al., 2006a] van Hee, K., et others (2006a). **Yasper: a tool for workflow modeling and analysis**. In *Proc. of the 6th Int. Conf. on Application of Concurrency to System Design (ACSD'06)*, pages 279–282, Turku, Finland. IEEE CS.

[van Hee et al., 2006b] van Hee, K., Oanea, O., Sidorova, N., et Voorhoeve, M. (2006b). **Verifying generalized soundness of workflow nets**. In *International Andrei Ershov Memorial Conference on Perspectives of System Informatics*, pages 235–247. Springer.

[Van Hee et al., 2003] Van Hee, K., Sidorova, N., et Voorhoeve, M. (2003). **Soundness and separability of workflow nets in the stepwise refinement approach**. In *ICATPN*, volume 2679, pages 337–356. Springer.

[Van Hee et al., 2004] Van Hee, K., Sidorova, N., et Voorhoeve, M. (2004). **Generalised soundness of workflow nets is decidable**. Springer.

[Van Hee et al., 2010] Van Hee, K. M., et Liu, Z. (2010). **Generating benchmarks by random stepwise refinement of petri nets.** In *ACSD/Petri Nets Workshops*, pages 403–417. Citeseer.

[Vanhatalo et al., 2007] Vanhatalo, J., Völzer, H., et Leymann, F. (2007). **Faster and more focused control-flow analysis for business process models through sese decomposition**. In *International Conference on Service-Oriented Computing*, pages 43–55. Springer.

[Verbeek et al., 2000] Verbeek, E., et Van Der Aalst, W. M. (2000). **Woflan 2.0 a petri-net-based workflow diagnosis tool**. In *International Conference on Application and Theory of Petri Nets*, pages 475–484. Springer.

[Verbeek et al., 2001] Verbeek, H. M., Basten, T., et van der Aalst, W. M. (2001). **Diagnosing workflow processes using Woflan**. *The computer journal*, 44(4):246–279.

[Vilkomir et al., 2001] Vilkomir, S., et Bowen, J. (2001). **Formalization of software testing criteria using the Z notation**. In *25th Int. Conf. on Computer Software and Applications (COMPSAC'01)*, pages 351–356, Chicago, IL, USA. IEEE CSP.

[Voorhoeve et al., 1997] Voorhoeve, M., et Van der Aalst, W. (1997). **Ad-hoc workflow: problems and solutions**. In *Database and Expert Systems Applications, 1997. Proceedings., Eighth International Workshop on*, pages 36–40. IEEE.

[West et al., 2001] West, D. B., et others (2001). **Introduction to graph theory**, volume 2. Prentice hall Upper Saddle River.

[Westergaard, 2011] Westergaard, M. (2011). **Better algorithms for analyzing and enacting declarative workflow languages using ltl**. In *International Conference on Business Process Management*, pages 83–98. Springer.

[White, 2004] White, S. A. (2004). **Introduction to bpmn**. *IBM Cooperation*, 2(0):0.

[Wimmel et al., 2011] Wimmel, H., et Wolf, K. (2011). **Applying cegar to the petri net state equation**. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 224–238. Springer.

[Yamaguchi et al., 2009] Yamaguchi, M., Yamaguchi, S., et Tanaka, M. (2009). **A model checking method of soundness for workflow nets**. *IEICE transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 92(11):2723–2731.

List of Figures

2.1 An ordinary Petri net in two states illustrating the firing of a transition 12
2.2 An generalised Petri net in two states illustrating the firing of a transition 13
2.3 State machine 16
2.4 Marked graph 16
2.5 Free choice net 17
2.6 Asymmetric choice net 17
2.7 Venn diagram of Petri nets subclasses 18
2.8 An example of coloured Petri net 20
2.9 Illustration of causal dependency, conflict and concurrency in workflow nets . . . 21
2.10 On-demand order delivery workflow net 22
2.11 Two strong trace equivalent workflow nets 23
2.12 Two weak trace equivalent workflow nets with respect to $\{t_2, t_4\}$ 24
2.13 Illustration of the place refinement of place p with a workflow net N_{sub} 26
2.14 Illustration of the transition refinement of transition t with a workflow net N_{sub} . . 27
2.15 Reduction rules of [Murata, 1989] 37
2.16 Reduction Rule ϕ_A of [Desel et al., 2005] 37

3.1 Workflow net (N_1) used to illustrate a spurious solution of the *state equation* . . . 47
3.2 Illustration of a state equation solution subnet ($sN(\nu_1)$) 48
3.3 Workflow net (N_2) used to illustrate a spurious solution of Q 50
3.4 Illustration of an execution modelled by three segments 52

4.1 Reduction rule $\phi_{RemoveP}$ (R_1) 69
4.2 Reduction rule $\phi_{RemoveT}$ (R_2) 71
4.3 Reduction rule $\phi_{RemoveST}$ (R_3) 72
4.4 Reduction rule $\phi_{RemoveTP}$ (R_4) 73
4.5 Reduction rule $\phi_{RemovePT}$ (R_5) 75
4.6 Reduction rule $\phi_{RemoveR}$ (R_6) 76
4.7 Illustration of the application of Algorithm 2 79

5.1 Excerpt of issue tracking system workflow net 90
5.2 Login and navigation coloured *sub*-workflow net 92
5.3 *sub*-CWF-nets of the *Question and Answer* CWF-net 93
5.4 *Payout Process* of the *Tax Accounting Manager* study case 96
5.5 Modal specification verification tool chain 98
5.6 An example of workflow net 100
5.7 SMT-Lib representation of a segment of workflow 101
5.8 Prolog representation of a segment of workflow 101
5.9 Input queries using respectively Z3 and SICStus 101
5.10 The three execution's segments proposed by both solvers 102
5.11 Workflow nets reduction tool 103
5.12 Example of the output of the execution of *Hadara-AdSimul-Red* 104
5.13 Architecture of the generation toolchain 108
5.14 A workflow net satisfying the *must*-formula $t_0 \wedge t_1 \wedge (t_2 \vee t_3 \vee t_4)$ 109
5.15 The extended workflow obtained from the workflow of Figure 5.14 109
5.16 Workflow nets generation parameters and their values 111
5.17 State-machine - may-valid and must-invalid modal specifications 113
5.18 State-machine - must-valid and may-invalid modal specifications 113
5.19 Marked-graph - may-valid and must-invalid modal specifications 114
5.20 Marked-graph - must-valid and may-invalid modal specifications 114
5.21 Free-choice - may-valid and must-invalid modal specifications 115
5.22 Free-choice - must-valid and may-invalid modal specifications 115
5.23 Ordinary workflow nets - may-valid and must-invalid modal specifications 116
5.24 Ordinary workflow nets - must-valid and may-invalid modal specifications 116
5.25 Reduction factors of workflow nets of *IBM-bpm* 120
5.26 Reduction factors of workflow nets of *SAP-ref* 120
5.27 Reduction time of workflow nets of the *gen1* 122
5.28 Reduction time of workflow nets of the *gen2* 123

7.1 Illustration of error-patterns 133

List of Tables

5.1 *Issue tracking system* requirements . . . 91
5.2 Colours of *Question and Answer* coloured workflow net's places . . . 92
5.3 Colours, inputs, outputs, and guards of *Question and Answer* coloured workflow net's transitions . . . 93
5.4 *Question and Answer* portal requirements . . . 95
5.5 Function associated with transitions of the *Payout Process* (Figure 5.4) . . . 96
5.6 Mean costs associated with transitions of the *Payout Process* . . . 97
5.7 *Payout Process* requirements . . . 97
5.8 *Issue Tracking System*: Experimentation results . . . 105
5.9 *Question and Answer Portal*: Experimentation results . . . 106
5.10 *Tax Accounting Manager*: Experimentation results . . . 107
5.11 Metrics over state-machine workflow nets . . . 112
5.12 Metrics over marked-graph workflow nets . . . 112
5.13 Metrics over free-choice workflow nets . . . 112
5.14 Metrics over ordinary workflow nets . . . 112
5.15 Reduction Results over the Issue Tracking System . . . 119
5.16 Results for the *IBM-bpm* and *SAP-ref* benchmarks . . . 121
5.17 Reduction results over the workflow nets of the Model Checking Contest . . . 122

List of Definitions

1 Definition: Cartesian Product . . . 10
2 Definition: Binary Relation . . . 10
3 Definition: Sequence . . . 10
4 Definition: Monoid . . . 10
5 Definition: Directed Graph . . . 11
6 Definition: Bipartite Directed Graph . . . 11
7 Definition: Path . . . 11
8 Definition: Ordinary Petri net . . . 11
9 Definition: Generalised Petri net . . . 13
10 Definition: Reachability Problem . . . 15
11 Definition: Boundedness . . . 15
12 Definition: Liveness . . . 15
13 Definition: State Machine . . . 16
14 Definition: Marked Graph . . . 16
15 Definition: Free Choice Net . . . 17
16 Definition: Asymmetric Choice Net . . . 17
17 Definition: Hierarchical Petri Nets . . . 19
18 Definition: Coloured Petri net . . . 19
19 Definition: Workflow net . . . 22
20 Definition: Strong Trace Equivalence . . . 23
21 Definition: Weak Trace Equivalence . . . 24
22 Definition: Soundness [van der Aalst, 1998, van der Aalst et al., 2011] . . . 25
23 Definition: k-soundness [Barkaoui et al., 2007] . . . 25
24 Definition: Generalised Soundness [Barkaoui et al., 2007] . . . 25
25 Definition: Hierarchical Workflow Nets . . . 26
26 Definition: Modal Workflow net . . . 28
27 Definition: T-invariant . . . 32
28 Definition: P-invariant . . . 32
29 Definition: Siphon . . . 33

30 Definition: Trap 33
31 Definition: Siphon-Trap Property 34
32 Definition: k-closure of a Workflow Net 34

33 Definition: Well-formed Modal Specification Formulae 45
34 Definition: Extended Modal Specifications 46
35 Definition: State Equation Constraint System 46
36 Definition: State Equation Solution Subnet 48
37 Definition: State Equation + Absence of Siphon Constraint System 49
38 Definition: State Equation + Absence of Siphon + MG Constraint System 50
39 Definition: k-segment Execution Constraint System 51
40 Definition: k-segment Modal Execution Constraint System 53
41 Definition: Abstract Petri Net 55
42 Definition: Marking of an Abstract Petri Net 56
43 Definition: Transition Instance of an Abstract Petri Net 56
44 Definition: Abstract Workflow Net 57
45 Definition: Well-formed Abstract Modal Specification Formulae 59
46 Definition: Extended Abstract Modal Specification Formulae 60
47 Definition: Abstract Extended Modal Specification 61
48 Definition: Subnet of a solution of $\mathcal{S}_a$ 62
49 Definition: State Equation + Absence of Siphon Constraint System 62
50 Definition: State Equation + Absence of Siphon Constraint + MG System 62
51 Definition: k-segment Abstract Execution 63
52 Definition: k-segment Abstract Execution 64

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — http://www.multiagent.fr/ThesisStyle
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/

Abstract:

Nowadays workflows are extensively used by companies and organisations in order to improve organizational efficiency, responsiveness and profitability by managing the tasks and steps of business processes. The verification of specifications has become mandatory to ensure that such processes are properly designed and reach the expected level of trust and quality. In this context, this thesis addresses the verification of modal specifications – *necessary* or *admissible* behaviour involving several activities and their causalities – of workflow nets – a Petri nets class suited for the description of workflows.
In particular, it defines an innovative constraint system based framework to model executions of ordinary as well as coloured workflow nets, and verify modal specifications. Further, it presents powerful reduction methods preserving properties of interest such as generalised soundness and correctness of a given modal specification. Such reduction methods are then portrayed as pre-processing steps reducing workflow nets size, so that the verification of preserved properties can be carried out on smaller instances.
Finally, as a practical contribution, this thesis introduces the tools that have been implemented as well as experimentations that have been carried out over industrial workflow nets in order to validate the approaches proposed in this thesis. The convincing experimental results highlight the effectiveness, efficiency and scalability of the modal specification verification method and reduction methods introduced in this thesis.

Keywords: Verification, Modal specification, Workflow net/Petri net, Constraint solving, Reduction

Résumé :

De nos jours, les workflows sont largement utilisés par les entreprises et les organisations en vue d'améliorer l'efficacité organisationnelle, la réactivité et la rentabilité en gérant les tâches et les étapes de processus opérationnels. La vérification des spécifications est devenue obligatoire afin d'assurer que ces processus sont correctement conçus et atteignent le niveau de confiance et de qualité attendu.
Dans ce contexte, cette thèse porte sur la vérification de spécifications modales – comportements nécessaires ou recevables impliquant plusieurs activités et leurs causalités – de workflows nets – une classe de réseaux de Petri adaptés à la description de workflows.
En particulier, cette thèse définit un cadre novateur permettant de modéliser les exécutions de workflow nets, avec ou sans données, et de vérifier des spécifications modales à l'aide de systèmes de contraintes. Elle présente également deux méthodes de réduction préservant la 'generalised soundness' et la validité d'une spécification modale donnée. Ces méthodes de réduction sont ensuite présentées comme des étapes de pré-traitement réduisant la taille des workflow nets, de sorte que la vérification des propriétés conservées puisse être effectuée sur de plus petites instances. Enfin, cette thèse présente les outils qui ont été mis en oeuvre ainsi que des expérimentations qui ont été menées sur un grand nombre de workflows industriels afin de valider les approches proposées dans cette thèse. Ces résultats expérimentaux convaincants mettent en évidence l'efficacité, l'efficience et le passage à l'échelle de la méthode vérification de spécification modales ainsi que des méthodes de réduction introduites dans cette thèse.

Mots-clés : Vérification, Spécifications Modales, Réseaux Workflows, Solveurs de Contraintes, Réduction

SPIM

■ École doctorale SPIM 16 route de Gray F - 25030 Besançon cedex
■ tél. +33 (0)3 81 66 66 02 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr

UFC
UNIVERSITÉ
DE FRANCHE-COMTÉ