# Approches de programmation par contraintes pour l'analyse des propriétés structurelles des réseaux de Petri et application aux réseaux biochimiques

Faten Nabli

université PARIS DIDEROT PARIS 7

UFR D'INFORMATIQUE DENIS-DIDEROT

# THESE

Présentée pour obtenir le titre de

## DOCTEUR DE L'UNIVERSITE PARIS DIDEROT

**Spécialté: Informatique**

Par

Faten Nabli

# Approches de programmation par contraintes pour l'analyse des propriétés structurelles des réseaux de Petri et application aux réseaux biochimiques

**Présentée le 10 juillet 2013 devant le jury composé de :**

| | | | |
|---|---|---|---|
| *Rapporteurs:* | Prof. Alexander Bockmayr | - | Freie Universität Berlin |
| | Prof. Karsten Wolf | - | Universität Rostock |
| *Superviseurs:* | Prof. François Fages | - | INRIA Paris-Rocquencourt |
| | Dr. Sylvain Soliman | - | INRIA Paris-Rocquencourt |
| *Président:* | Prof. Michel Habib | - | Université Paris Diderot- Paris7 |
| *Examinateurs:* | Prof. Hanna Klaudel | - | Université d'Evry |
| | Dr. Sabine Peres | - | Université Paris-Sud |
| *Invités :* | | - | |

# Contents

# List of Figures

# List of Tables

# Abstract

Petri nets are a simple formalism for modelling concurrent computation. This formalism has been proposed as a promising tool to describe and analyse biochemical networks. In this thesis, we explore the structural properties of Petri nets as a mean to provide information about the biochemical system evolution and its dynamics, especially when kinetic data are missing, making simulations impossible.

In particular, we consider the structural properties of siphons and traps. We show that these structures entail a family of particular stability properties which can be characterized by a fragment of CTL over infinite state structures. Mixed integer linear programs have been proposed and a state-of-the-art algorithm from the Petri net community has been described later to compute minimal sets of siphons and traps in Petri nets. We present a simple boolean model capturing these notions and compare SAT and CLP methods for enumerating the set of all minimal siphons and traps of a Petri net. Our methods successfully apply to a practical benchmark composed of the 404 models from the biomodels.net repository. We analyse why these programs perform so well on even very large biological models although the decision problem is NP-complete. We show that, in networks with bounded tree-width, the existence of a minimal siphon containing a given set of places can be decided in linear time, and the Siphon-Trap property as well.

Moreover, we consider two other Petri net structural properties: place and transition invariants. We present a simple method to extract minimal semi-positive invariants of a Petri net as a constraint satisfaction problem on finite domains using constraint programming with symmetry detection and breaking. This allows us to generalize well-known results about the steady-state analysis of symbolic Ordinary Differential Equations systems corresponding to biochemical reactions by taking into account the structure of the reaction network. The study of the underlying Petri net, initially introduced for metabolic flux analysis, provides classes of reaction systems for which the symbolic computation of steady states is possible.

# General Introduction

## Systems Biology

The development of high-throughput data collecting techniques, for example, micro-arrays, protein chips, yeast two-hybrid screens, etc., are used in systems biology to create databases, interaction maps and detailed models of complex reaction systems at the cellular level. As examples of large models that emerged over the past few decades, we can cite the biggest MOOSE (multi-scale) model [32] with about 7500 species and 10000 reactions; a large structural yeast model with 2153 species (1168 metabolites, 832 genes, 888 proteins and 96 catalytic protein complexes) and 1857 reactions (1761 metabolic reactions and 96 complex formation reactions); the RB/E2F map established by Curie Institute [10], with 530 reactions and 390 species, and being currently merged with the EGFR map of the Systems Biology Institute [82] with its 219 reactions and 322 species.

To facilitate exchange of such models, standard formats for encoding quantitative information have been developed. As examples of such formats, we find CellML (`www.cellml.org`) and NeuroML (`www.neuroml.org`), but the Systems Biology Markup Language (SBML) [54] has so far been the most successful standard model exchange format in this field. It has been adopted by more than 180 software systems ranging from simulators to model editors and databases.

It is also crucial to classify this enormous amount of models into databases where models can be freely deposited and distributed in standardised formats. New databases for systems biology appeared like the Reactome database (`www.reactome.org`) that contains curated models of core pathways and reactions in human biology. EcoCyc (`www.ecocyc.org`) and MetaCyc (`wwww.metacyc.org`) are databases of non redundant, experimentally elucidated metabolic pathways. EcoCyc contains data modelling metabolic pathways of the bacterium Escherichia coli K-12 MG1655. MetaCyc covers data from different organisms. It contains more than 1928 pathways from more than 2263 different organisms, and is curated from the scientific experimental literature. Biomodels.net (`www.biomodels.net`) database has also become a recognised reference repository for systems biology. It is a freely-accessible on line resource for storing, viewing, retrieving, and analysing published, peer-reviewed quantitative models of biochemical and cellular systems.

Simulation is currently the primary use of these models and the structure and

behaviour of each simulation model distributed by Biomodels.net database are thoroughly checked; in addition, model elements are annotated with terms from controlled vocabularies as well as linked to relevant data resources. Models can be examined on line or downloaded in various formats. Reaction network diagrams generated from the models are also available in several formats. Biomodels.net database also provides features such as on line simulation and the extraction of components from large scale models into smaller sub-models. Finally, the system provides a range of web services that external software systems can use to access up-to-date data from the database.

Having biochemical models and filling them with experimental data was the first challenge of biologists. With the increase of available information, the challenge of efficient information processing is arising: especially when kinetic parameters are lacking which make simulations not possible without guessing them. In some cases, the model structure may however be detailed enough to provide some useful information about the dynamics.

# Petri nets

Petri nets are a simple formalism for modelling concurrent computation. Petri nets provide a tool to model biochemical reaction networks and provide powerful qualitative and quantitative analysis techniques. The intuitive description of chemical process coupled with the possibility to simulate and analyse token movement, representing substance or information flow in biochemical systems, facilitates the use of Petri net in systems biology. The motivation for using Petri nets to model biochemical networks comes mainly from the fact that biochemical systems exhibit many concurrent reactions, similar to concurrent process in technical systems. In 1993, Reddy et al. introduced a method of representation of metabolic pathways as Petri nets, and illustrate some useful properties like liveness, reachability, reversibility fairness and invariants [85]. Then in 1994, Hofestadt [50] described the metabolic process depending on expressed genes as Petri nets. Moreover, Petri nets are also suitable to model medical systems because of the lack of experimental data due to experiments difficulties and ethical reasons. Since 1993, many applications have been published and Petri nets have been applied to many biochemical examples such as metabolic systems [50, 85, 107] gene regulatory networks [72, 73], signal transduction networks [73, 89] the glycolysis and pentose phosphate pathway and the carbon metabolism in Tuber. Different types of Petri nets extension have been used to model and analyse biochemical networks, going from coloured Petri nets, hybrid Petri nets [72, 73], continuous Petri nets[51] and stochastic Petri nets.

# Thesis contribution

In this thesis, we focus on Petri nets structural properties and how they can be computed using constraint programming techniques. The structural properties of interest give us some information about the biochemical reaction network and its dynamics, even when there is a lack in kinetic data making simulation impossible. As application, we evaluate them on the Biomodels.net database.

We consider the Petri net concepts of siphons and traps. A siphon is a set of places that, once it is unmarked, remains so. A trap is a set of places that, once it is marked, can never loose all its tokens. Thus, siphons and traps have opposing effects on the token distribution in a Petri net. These structural properties provide sufficient conditions for reachability (whether the system can reach a given state) and liveness (freedom of deadlocks) properties. It is proved that in order to be live, it is necessary that each siphon remains marked. Otherwise (i.e. once it is empty), transitions having their input places in a siphon can not be live. One way to keep each siphon marked is to have a marked trap inside it. In fact, this condition is necessary and sufficient for a free-choice net to be live [83]. Siphons can correspond to a set of metabolites that are gradually reduced during starvation whereas traps can correspond to accumulation of metabolites that are produced during the growth of an organism. It has been shown that finding a minimal siphon of a given cardinality or even containing a given set of places is NP-complete [101, 106]. Mixed integer linear programs have been proposed in [78, 21] and a state-of-the-art algorithm from the Petri net community has been described later in [23] to compute sets of minimal siphons and traps in Petri nets. We present a simple Boolean model capturing these notions and two methods for enumerating the set of all minimal siphons and traps of a Petri net. The first method iterates the resolution of the Boolean model executed with a SAT solver. The second method proceeds by backtracking with a CLP(B) program. On a benchmark composed of the 80 Petri nets of Petriweb [1] [45] and the 404 curated biological models of the biomodels.net [2] repository [66], we show that miniSAT and CLP(B) solvers are both faster by two orders of magnitude than the dedicated algorithms and can in fact solve all instances.

Furthermore, we analyse why these programs perform so well even on very large biological models. We show that in networks with bounded tree-width, the existence of a minimal siphon containing a given set of places can be decided in linear time [80]. We show also that deciding the Siphon-Trap property can be done in linear time in networks with bounded tree-width.

Moreover, we show that siphons and traps entail a family of particular stability properties which can be characterized by a fragment of CTL over infinite state structures. Interestingly, we show that a well-founded Boolean abstraction of the Petri net preserves a similar characterization of Boolean CTL properties as siphons and traps. This fragment of boolean CTL formulas can thus be verified without recourse to symbolic model-checking methods by a purely structural analysis of the

[1] http://www.petriweb.org/
[2] http://www.biomodels.net/

Petri net.

We also consider the Petri net structural properties of place and transition invariants. A P-invariant stands for a set of places over which the weighted some of tokens is constant and independent of any firing. P-invariants correspond to conservation law. On the other hand, a T-invariant has actually two interpretations in the given biochemical network: first, the entries of a T-invariant represent a multi-set of transitions, which reproduce a given marking by their partially ordered firing. That means that they occur basically one after the other. The partial order sequence of the firing events of the T-invariant's transitions may contribute to a deeper understanding of the system behaviour. Second, the entries of a T-invariant may also be read as the relative firing rates of transitions, all of them occurring permanently and concurrently. This activity level corresponds to the steady state behaviour and minimal T-invariants correspond to elementary modes classically studied for metabolic networks.

We present a simple method to extract minimal semi-positive invariants of a Petri net modelling a biological reaction system, as a constraint satisfaction problem on finite domains using constraint programming with symmetry detection and breaking. An implementation based on GNU-Prolog's FD solver of the method is incorporated in the BIOCHAM modelling environment [9, 35]. Two prototypes for computing minimal P-invariants and minimal T-invariants are evaluated on our benchmark of interest. Moreover, we present a way to generalize well-known results about the steady-state analysis of some symbolic Ordinary Differential Equations systems by taking into account the structure of the reaction network. The structural study of the underlying Petri net, usually used mostly for metabolic flux analysis, provides classes where the computation of some steady states of the system is possible, even though the original symbolic model did not form an S-system and was not solvable by state-of-the-art symbolic computation software.

# Roadmap

The manuscript is organized as follows. In the first chapter we focus on the Petri net formalism. We define notions and notations that are necessary for the understanding of the following work. We prove a new linear complexity result for the siphon extraction problem in Petri nets of bounded tree-width. We develop links between the structural properties of siphons and traps and their dynamical properties in Computation Tree Logic.

In the second chapter, we present the biological background, here we recall the concepts of molecular biology and we enlarge our definition of biochemical networks.

In the third chapter, we describe a boolean model for the problem of enumerating all minimal siphons in a Petri net and we compare two boolean methods to a state-of-the-art algorithm from the Petri net community [23].

In the last chapter, we introduce a new method based on Finite Domain con-

straint programming to efficiently compute place and transition invariants of a reaction network. It includes symmetry detection and breaking and scales up well to the biggest reaction networks found. We present also a new method to compute, in a fully analytical way, steady states of biochemical systems defined by a system of Ordinary Differential equations.

Finally, we conclude this manuscript with a summary and an outlook on future work.

# Chapter 1

# Petri Nets

## Contents



## Introduction

Petri nets have been introduced in 1962 by Carl Adam Petri in his dissertation. Petri nets are a mathematical modelling tool for describing and analysing systems characterized as being concurrent, parallel, distributed and/or stochastic. Since their early introduction, Petri nets and their concepts have been extended and developed, and applied in a variety of areas such as office automation, work-flows, flexible manufacturing, protocols and networks, real-time systems, embedded systems, telecommunications, Internet, e-commerce, trading and biological systems.

## 1.1 Definitions

A Petri net or place/transition net is a directed bipartite graph, in which the nodes represent transitions (i.e. events that may occur, graphically represented by rectangles) and places (i.e. conditions, graphically represented by circles). The directed arcs are allowed to connect only nodes from different classes. The dynamics of a Petri net is obtained by moving the tokens in the places (graphically represented by bullets inside places). A directed arc represents conditions to fire transitions. Formal definitions and basic terminology is presented in this section.

**Definition 1** (Petri net graph)**.** *A* Petri net graph *is a weighted bipartite directed graph $PN = (P, T, W)$, where $P$ is a finite set of vertices called* places*, $T$ is a finite set of vertices (disjoint from $P$) called* transitions *and $W : ((P \times T) \cup (T \times P)) \to \mathbb{N}$ represents a weight function attached to the arcs.*

The weight is represented by an integer located near the arc. If this integer is missing, it is assumed that the weight of the arc is 1 (the weight zero represents the absence of arc).

**Definition 2** (Marking)**.** *A* marking *for a Petri net graph is a mapping $m : P \to \mathbb{N}$ which assigns a number of tokens to each place. We say that a place $p$ is marked if and only if $m(p) > 0$, otherwise, it is said to be unmarked.*

**Definition 3** (Petri net)**.** *A* Petri net *is a 4-tuple $(P, T, W, m_0)$ where $(P, T, W)$ is a Petri net graph and $m_0$ is an* initial marking*.*

**Definition 4** (Predecessors/successors)**.** *The set of* predecessors *(resp.* successors*) of a transition $t \in T$ is the set of places ${}^\bullet t = \{p \in P \mid W(p,t) > 0\}$ (resp. $t^\bullet = \{p \in P \mid W(t,p) > 0\}$). Similarly, the set of predecessors (resp. successors) of a place $p \in P$ is the set of transitions ${}^\bullet p = \{t \in T \mid W(t,p) > 0\}$ (resp. $p^\bullet = \{t \in T \mid W(p,t) > 0\}$).*

*The set of predecessors (resp. successors) ${}^\bullet S$ (resp. $S^\bullet$) of a set of places $S$ is the union of sets of predecessors (resp. successors) of each place $p \in S$: ${}^\bullet S = \bigcup_{p \in S} {}^\bullet p$ (resp. $S^\bullet = \bigcup_{p \in S} p^\bullet$).*

*The set of predecessors (resp. successors) ${}^\bullet Q$ (resp. $Q^\bullet$) of a set of transitions $Q$ is the union of sets of predecessors (resp. successors) of each transition $t \in Q$: ${}^\bullet Q = \bigcup_{t \in Q} {}^\bullet t$ (resp. $Q^\bullet = \bigcup_{t \in Q} t^\bullet$).*

**Definition 5** (Transition enabling and firing)**.** *For every two markings $m, m' : P \to \mathbb{N}$ and every transition $t \in T$, there is a transition step $m \xrightarrow{t} m'$, if for all $p \in P$, $m(p) \geq W(p,t)$ and $m'(p) = m(p) - W(p,t) + W(t,p)$. A transition $t$ is enabled at marking $m$, ($m \xrightarrow{t}$) if and only if $\forall p \in {}^\bullet t : m(p) \geq W(p,t)$. $m \xrightarrow{t} m'$ means that the transition $t$ is enabled in $m$ and its firing leads to $m'$. $m'$ is such that $\forall p \in P, m'(p) = m(p) - W(p,t) + W(t,p)$. An enabled transition may or may not fire (depending on whether or not the corresponding event actually takes place).*

**Definition 6** (Firing sequence)**.** *A finite sequence of transitions $\sigma = (t_0 \dots t_n)$ is a finite* firing sequence *of the Petri net if there exists a sequence of markings $m_1, \dots, m_n$ for which $m \xrightarrow{t_0} m_1 \xrightarrow{t_1} \dots \xrightarrow{t_{n-1}} m_n \xrightarrow{t_n} m'$. This is denoted as $m \xrightarrow{\sigma} m'$.*

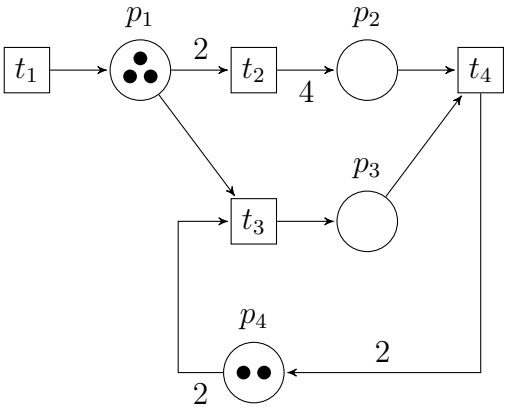



Figure 1.1: A Petri net with an arbitrary marking enabling $t_3$

**Example 1.** *In the Petri net depicted in Figure 1.1 , starting from a marking with 3 tokens in $p_1$ and 2 tokens in $p_4$, one can remove two tokens from $p_1$ to produce 4 tokens in $p_2$ (firing of $t_2$) and one can remove one token from $p_1$ and 2 tokens of $p_4$ and produce one token in $p_3$ (firing of $t_3$). And then remove one token from $p_2$ and one token from $p_3$ to produce two tokens in $p_4$ (firing of $t_4$). From this initial marking, firing just $t_3$ leads to the Petri net of Figure 1.2. Petri nets can intuitively model metabolic reactions by corresponding places to metabolites and transitions to reactions. Tokens inside places represent the number of molecules or the level of concentration of species. In the next chapter, we provide some examples of Petri nets modelling biochemical networks.*

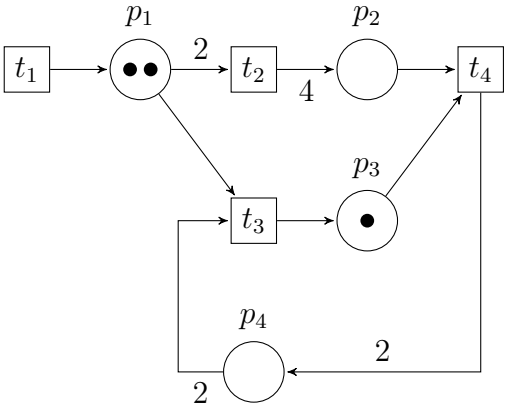



Figure 1.2: Petri net of Figure 1.1 after the firing of $t_3$

**Definition 7** (Incidence Matrix). *Let $P = (p_1, p_2, ..., p_n)$ (resp. $T = (t_1, t_2, ..., t_m)$) be the set of places (resp. transitions) of a Petri net. The incidence matrix of this Petri net is $A = [a_{ij}], i = 1, ..., n; j = 1, ...m$ defined as follows:*

$$a_{ij} = W(p_i, t_j) - W(t_j, p_i)$$

**Example 2.** *The Petri net of figure 1.1 has the following incidence matrix:*

$$\begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{bmatrix} 1 & -2 & -1 & 0 \\ 0 & 4 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -2 & 2 \end{bmatrix} \end{array}$$

*The second column of this matrix can be read as: the firing of $t_2$ removes two tokens from $p_1$ and produces four tokens in $p_2$.*

**Definition 8** (Firing count vector). *A vector $\bar{\sigma}$ is the firing count vector of $\sigma$ if $\bar{\sigma}(t)$ equals the number of times transition $t$ occurs in the firing sequence $\sigma$.*

**Definition 9** (Behaviour properties). *Let $N = (P, T, W, m_0)$ be a Petri net*

- *A place $p$ is $k$-bounded if there is a positive integer number $k$ such that the number of tokens in $p$ does never exceed $k$.*
- *A Petri net is $k$-bounded if all its places are $k$-bounded.*
- *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 $m_0$.*
- *A marking $m$ is dead if there is no transition enabled in $m$.*
- *A Petri net is deadlock free (weakly live) if there is no reachable dead markings.*
- *A Petri net is live (strongly live) if each transition is live.*

**Definition 10** (Net Structures). *Let $N = (P, T, W)$ a Petri net graph. $N$ is*

- *Homogeneous if $\forall p \in P : t, t' \in p^\bullet \Rightarrow W(p, t) = W(p, t')$.*
- *Ordinary if $\forall p \in P$ and $\forall t \in T$, $W(p, t) \leq 1$ and $W(t, p) \leq 1$.*
- *Extended simple (or asymmetric choice) if it is ordinary and $\forall p, q \in P : p^\bullet \cup q^\bullet = \emptyset \vee p^\bullet \subseteq q^\bullet \vee q^\bullet \subseteq p^\bullet$.*
- *Extended free choice if it is ordinary and $\forall p, q \in P : p^\bullet \cap q^\bullet = \emptyset \vee p^\bullet = q^\bullet$.*

## 1.2 Structural Properties

Various dynamical properties of Petri nets can be verified by constructing the reachability graph and analysing it. However, the reachability graph is generally infinite. In fact, the reachability problem was shown to be EXPSPACE-hard [68] years before it was shown to be decidable [74]. Structural analysis makes it possible to prove some properties without constructing the reachability graph. Structural properties do not depend on any marking: they depend only on the topology of the Petri net graph.

### 1.2.1 Place/Transition invariants

A place invariant is a set of places, whose weighted sum of tokens is constant independently of the sequence of firing. A transition invariant is a potential firing set without any net effect.

**Definition 11** (P-invariant)**.** *A vector $V = [v_1, v_2, ..., v_n]$ with non-negative integer components is a P-invariant if $V \neq 0$ and $VA = 0$, where $A$ is the incidence matrix of the Petri net with $n$ places and $m$ transitions.*

**Example 3.** *A P-invariant of the Petri net represented in figure 1.1 is such that:*

$$[v_1, v_2, v_3, v_4] \begin{bmatrix} 1 & -2 & -1 & 0 \\ 0 & 4 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -2 & 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

*which leads to:*

$$\begin{cases} v_1 & & & = 0 \\ -2v_1 + 4v_2 & & & = 0 \\ -v_1 & & +v_3 - 2v_4 & = 0 \\ & - v_2 & -v_3 + 2v_4 & = 0 \end{cases}$$

*Thus any vector $V = [0, 0, 2k, k]$, where $k$ is a non-negative integer is a p-invariant.*

**Proposition 1.** *[83] If $V$ is a P-invariant and $m_0$ is an initial marking of a Petri net, $Vm_0^t = Vm^t$ for any $m$ reachable from $m_0$.*

For instance, in the previous example, $2m(p_3) + m(p_4)$ is constant whatever $m$ is.

**Definition 12** (T-invariant)**.** *A vector $Y = [y_1, ...y_m]$ with non-negative integer components is a T-invariant if $AY^t = 0$, where $A$ is the incidence matrix of the Petri net with $n$ places and $m$ transitions.*

**Example 4.** *A t-invariant of a Petri net represented in figure 1 is such that:*

$$\begin{bmatrix} 1 & -2 & -1 & 0 \\ 0 & 4 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -2 & 2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

*which leads to:*

$$\begin{cases} y_1 - 2y_2 & -y_3 & & = 0 \\ 4y_2 & & - y_4 & = 0 \\ & y_3 & - y_4 & = 0 \\ & -2y_3 & + 2y_4 & = 0 \end{cases}$$

*Hence, any vector $Y = [6k, k, 4k, 4k]$, where $k$ is a non-negative integer is a t-invariant.*

**Proposition 2.** *[83] Let $\sigma$ be a firing sequence and $V_\sigma$ be the counting vector of $\sigma$. Let $m$ be the marking reached by firing $\sigma$. If $V_\sigma$ is a T-invariant, then $m = m_0$.*

**Definition 13** (Support)**.** *The* support *of a P-invariant $x$ or a T-invariant $y$ is the set of places or transitions respectively corresponding to the strictly positive components of $x$ and $y$, and are denoted by $\| x \|$ and $\| y \|$, i.e., $\| x \| = \{p \in P \mid x(p) > 0\}$ and $\| y \| = \{t \in T \mid y(t) > 0\}$.*

**Definition 14** (Minimal invariant)**.** *A P-invariant is minimal if no non-empty subset of the support is the support of some other P-invariant, i.e., $x$ is a minimal P-invariant if there is no other P-invariant $x'$ such that $x'(p) \leq x(p)$ for all $p$, and the greatest common divisor of all entries of $x$ is 1.*

*A T-invariant is minimal if no non-empty subset of the support is the support of some other T-invariant, i.e., $y$ is a minimal T-invariant if there is no other T-invariant $y'$ such that $y'(t) \leq y(t)$ for all $t$, and the greatest common divisor of all entries of $y$ is 1.*

### 1.2.2 Siphons and Traps

A siphon is a set of places that once unmarked remains so whereas a trap is a set of places that once marked remains so. We recall that a set of places $S$ is marked if at least one place $p$ in $S$ holds some tokens. Formal definitions are presented in the following. Let $PN = (P, T, W)$ be a Petri net graph.

**Definition 15.** *A* trap *is a non-empty set of places $P' \subseteq P$ whose successors are also predecessors: $P'^{\bullet} \subseteq {}^{\bullet}P'$.*

*A* siphon *is a non-empty set of places $P' \subseteq P$ whose predecessors are also successors: ${}^{\bullet}P' \subseteq P'^{\bullet}$.*

*A siphon (resp. trap) is* proper *if its predecessors set is strictly included in its successors set,* ${}^\bullet P' \subsetneq P'^\bullet$.

*A siphon (resp. a trap) is* minimal *if it does not contain any other siphon (resp. trap).*

It is worth remarking that a siphon in a Petri net graph $PN$ is a trap in the dual Petri net graph, obtained by reversing the direction of all arcs in $PN$. Note also that since predecessors and successors of an union are the union of predecessors (resp. successors), the union of two siphons (resp. traps) is a siphon (resp. a trap).

**Example 5.** *In the Petri net graph depicted in Figure 1.3,* $\{A, B\}$ *is a minimal proper siphon:* ${}^\bullet\{A, B\} = \{r_1, r_2\} \subset \{A, B\}^\bullet = \{r_1, r_2, r_3\}$. $\{C, D\}$ *is a minimal proper trap:* $\{C, D\}^\bullet = \{r_4, r_5\} \subset {}^\bullet\{C, D\} = \{r_3, r_4, r_5\}$.

The following propositions show that traps and siphons provide a structural characterization of some particular dynamical properties on markings.

**Proposition 3.** *[83] For every subset* $P' \subseteq P$ *of places,* $P'$ *is a trap if and only if for any marking* $m \in \mathbb{N}^P$ *with* $m_p \geq 1$ *for some place* $p \in P'$*, and any marking* $m' \in \mathbb{N}^P$ *such that* $m \xrightarrow{\sigma} m'$ *for some sequence* $\sigma$ *of transitions, there exists a place* $p' \in P'$ *such that* $m'_{p'} \geq 1$.

**Proposition 4.** *[83] For every subset* $P' \subseteq P$ *of places,* $P'$ *is a siphon if and only if for any marking* $m \in \mathbb{N}^P$ *with* $m_p = 0$ *for all* $p \in P'$*, and any marking* $m' \in \mathbb{N}^P$ *such that* $m \xrightarrow{\sigma} m'$ *for some sequence* $\sigma$ *of transitions, we have* $m'_{p'} = 0$ *for all* $p' \in P'$.

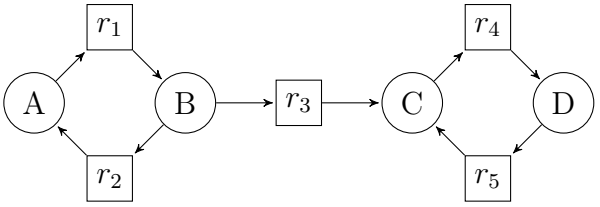



Figure 1.3: Petri net graph of Example 5.

Although siphons and traps are stable by union, it is worth noting that minimal siphons do not form a generating set of all siphons. A siphon is called a *basis* siphon if it can not be represented as a union of other siphons [78]. Obviously, a minimal siphon is also a basis siphon, however, not all basis siphons are minimal. For instance, in Example 5, there are two basis siphons, $\{A, B\}$ and $\{A, B, C, D\}$, but only the former is minimal, the latter cannot be obtained by union of minimal siphons.

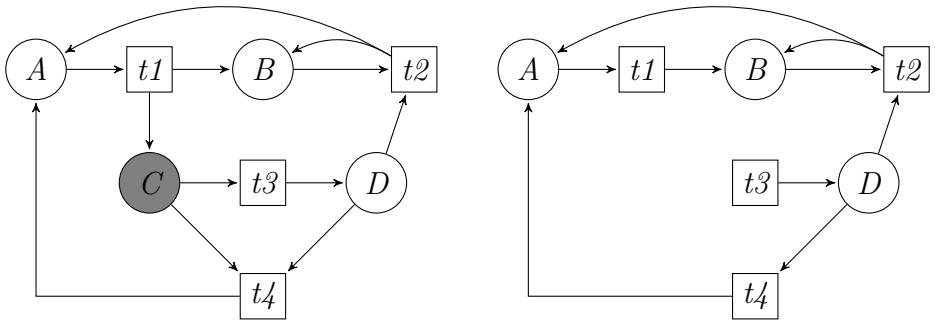
### Minimal Siphons and strong connectedness

In this part, we show that the set of places forming a minimal siphon (in term of inclusion) form a strongly connected subnet.

**Definition 16** (Strongly connected Petri net)**.** *A Petri net is strongly connected when for every pair of nodes (i.e., places and transitions) x and y, there exists a path leading from x to y.*

**Definition 17** (Subnet)**.** *Let G=(P, T, F) be a Petri net.* $\tilde{P} \subseteq P$ *and* $\tilde{T} \subseteq T$*. Then* $\tilde{G} = (\tilde{P}, \tilde{T}, \tilde{F})$ *is the subnet of G generated by* $\tilde{P} \cup \tilde{T}$ *iff* $\tilde{F}(p,t) = F(p,t)$*,* $\tilde{F}(t,p) = F(t,p)$ $\forall p \in \tilde{P}, \forall t \in \tilde{T}$*.*

**Example 6.** *The Petri net* $G(P,T,F)$ *is shown on figure on the left. The subnet generated by* $\{A, B, D\} \cup T$ *is shown on figure on the right.*




**Proposition 5.** *Let S be a minimal siphon in a Petri net N. The subnet* $N_S$ *generated by* $S \cup {}^\bullet S$ *is strongly connected.*

*Proof.* First we remark that every transition t of $N_S$ is a predecessor of some place of $N_S$ by definition of $N_S$ and also a successor of some place of $N_S$ since $S$ is a siphon. Hence, for proving that $N_S$ is strongly connected, we need just to show that for every two places $x$ and $y$, there is a path between $x$ and $y$ in $N_S$. Let $y \in S$ and let the set $X = \{z \in S \mid$ there is a path leading from z to y in $N_S\}$. It suffices to show that $X = S$. For every transition $t \in {}^\bullet y$, there exists some $y'$ in $S$ such that $t \in y'^\bullet$. Hence there exists a path leading from $y'$ to $y$. and ${}^\bullet t \cap S \subseteq X$ and ${}^\bullet X \subseteq X^\bullet$ then $X$ is a non empty siphon included in $S$. Since $S$ is minimal then $X = S$. □

### Application to the Siphon-Trap property

The concepts of live Petri nets (no dead transition) and especially of deadlock free Petri nets are important. A deadlock occurs if no transitions can be fired any longer. This corresponds to a system which is either badly designed or badly modelled. One reason to consider minimal siphons is that they provide a sufficient condition for the non-existence of deadlocks. It has been shown indeed that in a deadlocked Petri net (i.e. where no transition can fire), all unmarked places form a siphon [16]. The siphon-based approach for deadlocks detection checks if the net contains a proper siphon that can become unmarked by some firing sequence. A proper siphon does not become unmarked if it contains an initially marked trap. If such a siphon is identified, the initial marking is modified by the firing sequence and the check continues for the remaining siphons until a deadlock is identified, or until no further

progress can be done. Considering only the set of minimal siphons is sufficient because if any siphon becomes unmarked during the analysis, then at least one of the minimal siphons must be unmarked. The relevance of siphons and traps for other liveness properties are summarized in [47].

**Definition 18** (Siphon-Trap Property (STP)). *Given $PN = (P, T, W, m_0)$ a Petri net. The STP holds when every siphon includes a marked trap.*

**Theorem 1.** *[47] The following properties hold.*

1. *An ordinary Petri net without siphons is live.*
2. *An ordinary Petri net in which the STP holds is deadlock free.*
3. *An Extended Simple Petri net in which the STP holds is live.*
4. *An Extended Free Choice Petri net is live iff the STP holds.*

An additional interesting property is the monotonic liveness defined as follows:

**Definition 19** (Monotonic liveness). *[48] Let $PN = (P, T, W, m_0)$ a Petri net. PN is called monotonically live, if being live for $m_0$, it remains live for any $m' \geq m_0$.*

In [48], an interesting study of structural properties that preserve liveness is done. Two cases are distinguished: ordinary and non-ordinary nets. For ordinary nets, the following theorems are proved.

**Definition 20.** *Mono-T-semiflow (MTS) Petri net[48]*

*Let $N = (P, T, W)$ be a Petri net graph.*

*A net is conservative if every place belongs to the support of a P-invariant; a net is consistent if every transition belongs to the support of a T-invariant. A mono-T-semiflow (MTS) Petri net is a consistent and conservative net that has exactly one minimal T-invariant.*

**Theorem 2.** *[48] Let $PN$ be an ordinary Petri net graph. If $(PN, m_0)$ is monotonically live then the STP holds.*

**Theorem 3.** *[48] Let $PN$ be an ordinary mono-T-semiflow Petri net which for $m_0$ fulfils the STP. Then the system $(PN, m)$ is live for any $m \geq m_0$.*

The second case concerns non-ordinary nets, a transformation from non-ordinary to ordinary Petri nets that preservers monotonicity of liveness is used and the property of liveness is studied on ordinary nets. Application of this property on cases from biochemical networks is hold to demonstrate the helpfulness of the STP for biochemical networks. The essential analysis results show that all biochemical models hold the STP, they are consistent and live.

### 1.2.3 Known Time Complexity Of Minimal Siphon Extraction Problem

In this section, we review the time complexity results of minimal siphon extraction problem in general Petri nets.

**Definition 21** (isMinimal)**.** *The problem isMinimal is the following decision problem.*

*Input: a Petri net $N$ and a subset $S$ of places of $N$.*

*Output: is $S$ a minimal siphon of $N$.*

**Theorem 4.** *[101] The decision problem isMinimal is polynomial.*

**Definition 22** (Minimal Siphon Extraction Problem (MSEP))**.** *The problem MSEP is the following problem.*

*Input: A Petri net $N$ and a specified subset $Q$ of places of $N$.*

*Output: Find a minimal siphon $S$ containing $Q$.*

**Definition 23** (Rec-MSEP)**.** *Given a Petri net $N$ and a subset of places $Q$, Rec-MSEP is the following decision problem: "does there exist a minimal siphon of $N$ containing $Q$?".*

**Definition 24.** *(3-Satisfiability Problem (3-SAT))*

*Input: A set $V$ of variables and a collection $C$ of clauses (set of literals) over $V$ such that each clause $c \in C$ has exactly 3 literals.*

*Output: Is there a satisfying truth assignment for $C$?*

**Theorem 5.** *[106] Rec-MSEP is NP-complete.*

*Proof.* Rec-MSEP is NP-hard by polynomial reduction from 3-SAT [106] and NP-complete since IsMinimal is polynomial [101]. □

In [106], the NP-hardness of Rec-MSEP is proved by a polynomial reduction from 3-SAT. It is worthwhile resuming the proof for general Petri nets. For that, the following definitions are needed.

**Definition 25** (Q-hitting-siphon)**.** *Given a Petri net $N$ and a siphon $S$ of $N$, $S$ is called a Q-hitting-siphon if and only if any siphon $S' \subseteq S$ of $N$ includes $Q$.*

**Proposition 6.** *[106] There exists a minimal siphon containing $Q$ if and only if there exists a Q-hitting-siphon.*

**Definition 26** (Q-Hitting Siphon Extraction Problem (Q-HSE))**.** *The Q-HSE is the following problem.*

*Input: A Petri Net $N$ and a specified subset of places $Q$.*

*Output: Find a Q-hitting-siphon.*

**Definition 27** (Rec-Q-HSE)**.** *Rec-Q-HSE is the following decision problem:"given a Petri net graph $N = (P, T, W)$ and a subset of places $Q \subseteq P$, does there exist a $Q$-hitting siphon in $N$?".*

**Proposition 7.** *[106] The 3-SAT problem can be polynomially reduced to Rec-Q-HSE. There exists a satisfying truth assignment for the clauses if and only if $N$ has a $Q$-hitting siphon.*

*Proof.* The 3-SAT Reduction is illustrated in the following example.

**Example 7.** *Let $V = \{v_1, v_2, v_3, v_4\}$ and $C = \{c_1, c_2, c_3\}$ where*

$$c_1 = (v_1, v_2, v_3)$$
$$c_2 = (\bar{v_1}, v_3, v_4)$$
$$c_3 = (\bar{v_2}, \bar{v_3}, \bar{v_4})$$

*The corresponding Petri net $N = (P, T, E)$ is depicted in Figure 1.4 where $Q = \{q_0\}$.*

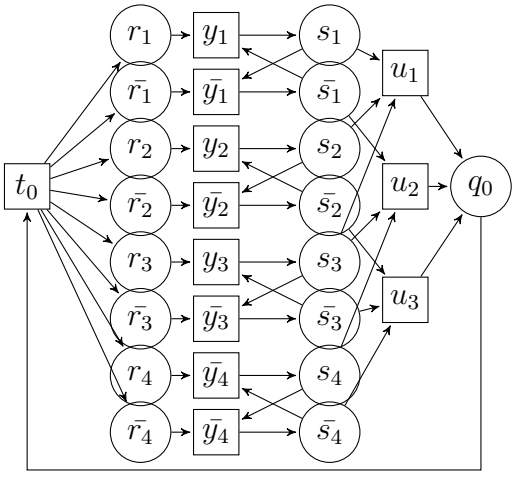



Figure 1.4: Petri net for reduction of 3-SAT problem of example 7: for $i = 1, 2, 3, 4$, $r_i$ and $\bar{r_i}$, $s_i$ and $\bar{s_i}$, $y_i$ and $\bar{y_i}$ correspond to literals, respectively variables and their negation. for $i = 1, 2, 3$, $u_i$ correspond to clauses.

More generally, given any instance of 3-SAT: $V = \{v_1, v_2, ..., v_n\}$, $\bar{V} = \{\bar{v_1}, \bar{v_2}, ..., \bar{v_n}\}$ $(n \geq 2)$ and $C = \{c_1, c_2, ...c_m\}$ $(m \geq 2)$, we construct an instance $N = (P, T, E)$ and $Q \subseteq P$ of Q-HSE such that:

$$P = Q \cup R \cup S$$

where:

$$Q = \{q_0\}$$
$$R = \{r_i \mid v_i \in V, 1 \leq i \leq n\} \cup \{\bar{r_i} \mid \bar{v_i} \in \bar{V}, 1 \leq i \leq n\}$$

$$S = \{s_i \mid v_i \in V, 1 \leq i \leq n\} \cup \{\bar{s_i} \mid \bar{v_i} \in \bar{V}, 1 \leq i \leq n\}$$

and $T = T_0 \cup Y \cup U$ where

$$T_0 = \{t_0\}$$
$$Y = \{y_i \mid v_i \in V, 1 \leq i \leq n\} \cup \{\bar{y_i} \mid \bar{v_i} \in \bar{V}, 1 \leq i \leq n\}$$
$$U = \{u_j \mid c_j \in C, 1 \leq j \leq m\}$$

- Let $\Psi : V \cup \bar{V} \to S$ be a bijection such that:

$$\Psi(l_i) = s_i \quad \text{if } l_i \in V$$
$$\Psi(l_i) = \bar{s_i} \quad \text{if } l_i \in \bar{V}$$
$$(l_i \text{ denotes } v_i \text{ or } \bar{v_i})$$

$$E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6$$
$$E_1 = \{(t_0, r_i), (t_0, \bar{r_i}) \mid 1 \leq i \leq n\}$$
$$E_2 = \{(r_i, y_i), (\bar{r_i}, \bar{y_i}) \mid 1 \leq i \leq n\}$$
$$E_3 = \{(y_i, s_i), (\bar{y_i}, \bar{s_i}), (s_i, \bar{y_i}), (\bar{s_i}, y_i) \mid 1 \leq i \leq n\}$$
$$E_4 = \{(\Psi(l_i), u_j) \mid l_i \in c_j, 1 \leq i \leq n, 1 \leq j \leq m\}$$
$$E_5 = \{(u_j, q_0) \mid 1 \leq j \leq m\}$$
$$E_6 = \{(q_0, t_0)\}$$

□

**Lemma 1.** *[106] Rec-Q-HSE for a general Petri-Net is NP-hard.*

*Proof.* There exists a satisfying truth assignment for $C$ if and only if $N$ has a Q-hitting-siphon. In deed, let $D \subseteq P$ be a Q-hitting-siphon of $N$ and let us prove that there exists a satisfying truth assignment for $C$. We have $q_0 \in D$, then, either $s_i$ or $\bar{s_i}$ in $D$ (for $i = 1, ..., n$, $\{s_i, \bar{s_i}\}$ is a minimal siphon included in $D$ and not containing $q_0$, which is in contradiction with the fact that $D$ is a Q-hitting-siphon). We have $q_0 \in D$ then, for $j = 1, ..., m$, $D \cap {}^\bullet u_j \neq \emptyset$. Define a mapping $\tau : V \cup \bar{V} \to \{true, false\}$ such that:

$$\tau(v_i) = \begin{cases} true & if s_i \in D, \\ false & if \bar{s_i} \in D, \\ true & if \{s_i, \bar{s_i}\} \cap D = \emptyset. \end{cases}$$

It is clear that $\tau$ is a satisfying truth assignment for $C$.

Conversely, suppose that there exists a satisfying truth assignment $\tau : V \rightarrow \{true, false\}$ for $C$, let us prove that there exists a Q-hitting-siphon of $N$. A Q-hitting-siphon is $D = S' \cup R' \cup \{q_0\}$ such that: $S' = \{s_i \in S \mid \tau(v_i) = true\} \cup \{\bar{s_i} \in S \mid \tau(v_i) = false\}$ and $R' = R \cap {}^\bullet({}^\bullet S')$. It is easy to see that $D$ is a siphon: for each $k \in \{1, .., m\}$, there is at least one $s'_j \in S'$ such that $|{}^\bullet u_k \cap \{s_j, \bar{s_j}\}| = 1$. Then ${}^\bullet q_0 \subset D^\bullet$. For each $s'_j \in S'$ then ${}^\bullet s'_j \in D^\bullet$ (${}^\bullet s'_j = \{y'_j\}$ and $r'_j \in D$). For each $r'_i \in R'$, ${}^\bullet r'_i = \{t_0\}$ and $\{t_0\} = q_0^\bullet$. Hence, ${}^\bullet D \subseteq D^\bullet$. Let $D'$ be any minimal siphon included in $D$, $D'$ includes either $s_j$ or $s'_j$ for $j = 1, ..., n$ then $D \cap R' \neq \emptyset$ and $q_0 \in D'$. Then, $D$ is a Q-hitting-siphon.

□

We consider the problem of existence of a siphon, not necessary minimal, of a given cardinality.

**Definition 28** (k-Siphon problem)**.** *The k-Siphon problem is the following decision problem.*

*Input: A Petri-net N and a positive integer k.*

*Output: Does there exist a siphon of cardinality k.*

**Definition 29** (k-set-covering)**.** *The k-set-covering is the following decision problem.*

*Input:*

- $\mathcal{U}$ *a finite set (the universe)*
- $\mathcal{S}$ *a subset of* $\mathcal{P}(\mathcal{U})$
- *an integer k*

*Output: Does there exist* $S \subseteq \mathcal{S}$ *of cardinality k such that* $\mathcal{U} = \bigcup S$

**Theorem 6.** *The k-Set-Covering decision problem is NP-complete. The optimization problem, consisting in finding the minimal k such that k-Set-Covering holds, is thus NP-hard.*

**Proposition 8.** *There is a polynomial reduction from k-Set-Covering to k-Siphon.*

*Proof.* Here, we provide a reduction from set-covering to k-Siphon problem:

Let N=(P, T, W) be the Petri-net such that

- Places are $\mathcal{S}$
- Transitions are $\mathcal{U}$
- $\forall t \in \mathcal{U}$, $t^\bullet = P$ and ${}^\bullet t = \{s \in \mathcal{S} \mid t \in s\}$

Then, for every subset $S$ of places ${}^\bullet S \subseteq S^\bullet \iff \mathcal{U} = \bigcup S$. In deed, let $S$ a subset of places, we have ${}^\bullet S = \mathcal{U}$. ${}^\bullet S \subseteq S^\bullet$ if and only if for any transition $u \in \mathcal{U}$, there exists $s \in S$ such that $u \in s$.

□

**Theorem 7.** *k-Siphon is:*

- *NP-hard (polynomially be reduced to the set-covering problem).*
- *NP-complete since verifying that a given set of places is a siphon of size k is linear.*

The related optimization problem, consisting in finding a minimal siphon of minimum cardinality, is thus NP-hard.

## 1.3 New Time Complexity Result

In this section, we present two complexity results which came from the analysis of the surprising good evaluation results of the third chapter. We show that the decision problem Rec-MSEP can be decided in polynomial time for Petri nets with bounded cut-width. This complexity result can be improved thanks to the application of Courcelle's theorem: we show that, given a parameter $k > 0$, Rec-MSEP can be decided in linear time for Petri nets with tree-width bounded by $k$. We prove also that given a parameter $k > 0$, deciding the Siphon Trap Property can be done in linear time for Petri nets with tree-width bounded by $k$. This complexity result was proved by Thierry Martinez [80].

### 1.3.1 Polynomial time complexity theorem for Petri-nets with bounded tree-width

We start by introducing the notions of cut-width and tree-width.

**Definition 30.** *(Cut-width)[62] Given a non-oriented graph $G = (V, E)$, $|V| = n$, a numbering of $G$ is a one-to-one mapping $L_G : V \to \{0, \ldots, n-1\}$. The cut-width of a numbering $L_G$ is*

$$\max_{0 \leq p < n} |\{\{u, v\} \in E : L_G(u) \leq p < L_G(v)\}|$$

*The cut-width $\mathrm{c}(G)$ of $G$ is the minimum cut-width over all the numberings.*

**Example 8.** *Figure 1.6 shows a numbering of the graph depicted in Figure 1.5. The numbering l corresponds to the ordering (C, A, D, E, B, F). In Figure 1.6, vertices are arranged in a line with the order of the numbering: C is located in the first position followed by vertex A and so on. The cut-width of each vertex is*

*represented as a dashed line with its corresponding value. For example, the cut-width of vertex C is $c_l(C) = 1$, because only the edge (C,B) has an endpoint in C labelled with 1 and the other endpoint in a vertex labelled with a value larger than 1. In a similar way, the cut-width of A equals 4, by counting the appropriate number of edges ((C,B), (A,B), (A,E), and (A,D)). Then, since the cut-width of the graph G, $c_l(G)$, is the maximum of the cut-width of its vertices, in this particular example we obtain $c_l(G) = c_l(D) = 5$.*

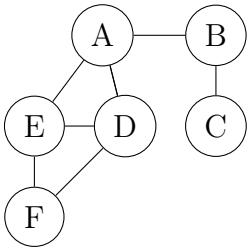



Figure 1.5: Graph example (G) with 6 vertices

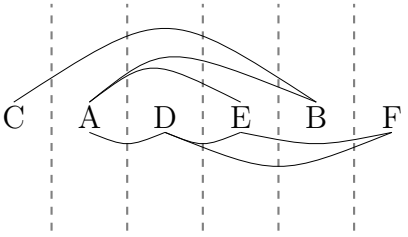



Figure 1.6: Cut-width of G for a numbering l

**Definition 31.** *(Tree decomposition)[86] A tree decomposition of a non-oriented graph $G = (V, E)$ is a pair $(X, T)$ where $T = (I, A)$ is a tree, and $X = \{X_i : i \in I\}$ is a family of subsets of $V$, such that*

1. $\bigcup_{i \in I} X_i = V$,
2. *Every edge of $G$ has both its ends in some $X_i$ $(i \in I)$,*
3. *For all $i, j, k \in I$, if $j$ lies on the path from $i$ to $k$ in $T$, then $X_i \cap X_k \subseteq X_j$.*

*The tree-width of a tree decomposition is $\max_{i \in I} |X_i| - 1$. The tree-width $\mathrm{tw}(G)$ of $G$ is the minimum tree-width taken over all possible tree decompositions of $G$.*

**Example 9.** *A tree decomposition of the graph shown in figure 1.5 is depicted in figure 1.7. The graph is decomposed onto a tree with 5 nodes. Each graph edge connects two vertices that belongs to some tree node. Graph vertices are adjacent only when the corresponding sub-trees intersect. Each tree node lists at most three vertices, hence the width of this decomposition is two.*

**Theorem 8** ([7])**.** *For all graph $G$, $\mathrm{tw}(G) \leq \mathrm{c}(G)$.*

**Definition 32** ([44])**.** *The primal graph of a hyper-graph $\mathcal{H} = (V, H)$ is the graph $G = (V, E)$ such that $E = \{\{X, Y\} \subseteq V \mid \exists h \in H, \{X, Y\} \subseteq h\}$.*

**Example 10.** *Figure 1.8 shows an example of a hyper-graph with $X = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $H = \{h_1, h_2, h_3, h_4\} = \{\{v_1, v_2, v_3\}, \{v_2, v_3\}, \{v_3, v_5, v_6\}, \{v_4\}\}$. The primal graph of $H$ is shown in figure 1.9.*

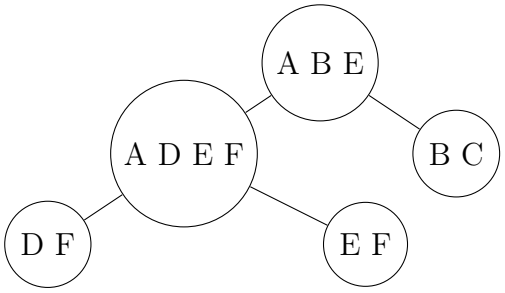



Figure 1.7: Tree decomposition G

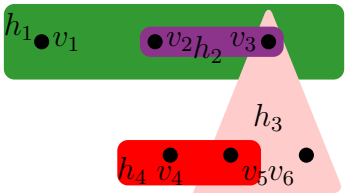



Figure 1.8: Hyper-graph example H

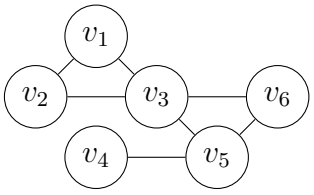



Figure 1.9: Primal graph of H

**Definition 33.** *(Cut-width and tree-width of a hyper-graph)[44] The cut-width and the tree-width of a hyper-graph are the cut-width and the tree-width respectively of its primal graph plus one.*

**Definition 34.** *(Constraint Satisfaction Problem (CSP)) A CSP instance is a triple* $(V, D, C)$ *where:*

- $V$ *is an ordered set of* $n$ *variables (*$v_i$ *is the* $i^{th}$ *element of* $v$*).*
- $D$ *is a mapping from* $V$ *to a set of domains* $\{d(v_1), d(v_2), .., d(v_n)\}$*. For each variable* $v_i \in V$ *,* $d(v_i)$ *is the finite domain of its possible values.*
- $C = \{c_1, c_2, .., c_m\}$ *is a set of* $m$ *constraints. Each constraint* $c_i \in C$ *is defined as a pair* $(vars(c_i), rel(c_i))$*, where:*
  - $vars(c_i) = (v_{i1}, .., v_{ik})$ *is an ordered subset of* $V$ *called the constraint scope.*
  - $rel(c_i)$ *is a subset of the Cartesian product* $d(v_{i1}) * .. * d(v_{ik})$ *and it specifies the allowed combinations of values for the variables in* $vars(c_i)$*.*

**Definition 35.** *(Cut-width and tree-width of a constraint satisfaction problem)[44] The cut-width and the tree-width of a constraint satisfaction problem are the cut-width and the tree-width respectively of its constraint hyper-graph.*

**Theorem 9.** *[44] Given a constraint satisfaction problem* $P$ *and a tree decomposition* $D$ *of* $P$*,* $P$ *can be solved in* $\mathbf{O}(|P|^{tw(D)+1} \cdot \log |P|)$*.*

**Definition 36.** *(Cut-width and tree-width of a Petri net) The cut-width and the tree-width of a Petri-net* $N = (P, T, w)$ *are the cut-width and the tree-width respectively of its underlying hyper-graph* $\mathcal{H}_N = (P, H)$ *with* $H = \{\{p \in P \mid w(p,t) \neq 0 \text{ or } w(t,p) \neq 0\} \mid t \in T\}$*.*

**Definition 37.** *Given a Petri net $N = (P, T, w)$ such that $P = \{p_0, \ldots, p_{n-1}\}$, $n \geq 2$, the constraint satisfaction problem* $\mathrm{CSP}(N) = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ *for finding a siphon of a parametric size $K$ in $N$ has the following constraints $\mathcal{C}$ on the variables $\mathcal{V} = \{X_0, S_0, \ldots, X_{n-1}, S_{n-1}, K\}$.*

- *for all* $0 \leq i < n$, $X_i = 1 \Rightarrow \bigwedge_{t \in {}^\bullet p_i} \bigvee_{p_j \in {}^\bullet t} X_j = 1$,
- $S_0 = X_0 + X_1$ *and for* $1 \leq i < n-1$, $S_i = S_{i-1} + X_{i+1}$,
- $S_{n-1} = K$

Given a Petri net $N$, definition 37 provides an encoding of the problem of finding a siphon of size $K$ in such a manner that we are able to bound the cut-width of $\mathrm{CSP}(N)$ by the cut-width of $N$, we decompose the constraint $\sum_{i=0}^{n-1} X_i = K$ into $n$ binaries constraints as illustrated in Example 11.

**Example 11.** *Consider Petri net $N$ depicted in Figure 1.10. The corresponding hyper and primal graphs are shown respectively in Figure 1.11 and Figure 1.12.*

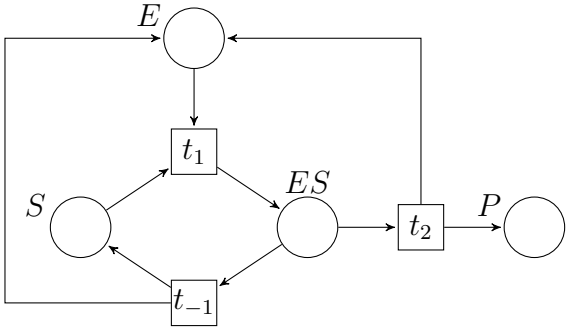



Figure 1.10: Petri net depicting the enzymatic reaction

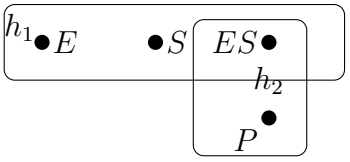



Figure 1.11: Hyper-graph of the Petri net depicted in Figure 1.10

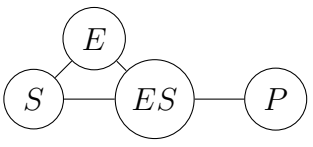



Figure 1.12: Primal graph of the Petri net depicted in Figure 1.10

*Boolean variables $X_0$, $X_1$, $X_2$ and $X_3$ are associated respectively to $E$, $S$, $ES$ and $P$. The following* $\mathrm{CSP} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ *encodes the problem of finding a siphon of $N$ of size $K$:*

$\mathcal{V} = \{X_0, S_0, X_1, S_1, X_2, S_2, X_3, S_3, K\}$

*The siphon constraints are:*

$$X_0 = 1 \Rightarrow X_2 = 1$$
$$X_1 = 1 \Rightarrow X_2 = 1$$
$$X_2 = 1 \Rightarrow X_0 = 1 \vee X_1 = 1$$
$$X_3 = 1 \Rightarrow X_2 = 1$$

*Binary constraints encoding the sum are:*

$$S_0 = X_0 + X_1$$
$$S_1 = S_0 + X_2$$
$$S_2 = S_1 + X_3$$
$$S_3 = K$$

**Lemma 2.** *For all Petri net $N$, $\mathrm{c}(\mathrm{CSP}(N)) \leq \mathrm{c}(N) + 2$.*

*Proof.* Suppose that $P = \{p_0, \ldots, p_{n-1}\}$ is enumerated such that $L_N : p_i \mapsto i$ is a numbering of the primal graph of the underlying hyper-graph of $N$ with $\mathrm{c}(L_N) = \mathrm{c}(N)$. Then the numbering $L_{\mathrm{CSP}(N)}$ such that for all $0 \leq i < n$, $L_{\mathrm{CSP}(N)}(X_i) = 2 \cdot i$, $L_{\mathrm{CSP}(N)}(S_i) = 2 \cdot i + 1$ and $L_{\mathrm{CSP}(N)}(K) = 2 \cdot n$ is such that $\mathrm{c}(L_{\mathrm{CSP}(N)}) = \mathrm{c}(N) + 2$. □

**Example 12.** *Consider again the Petri net $N$ of Figure 1.10 and the numbering $L_N$ corresponding to the ordering $(E, S, ES, P)$,we have $\mathrm{c}_{L_N}(N) = 3$. The primal graph $P$ corresponding to $\mathrm{CSP}(N)$ as defined in 37 is depicted in Figure1.13. On $P$, considering the numbering $L_{\mathrm{CSP}(N)}(X_i) = 2 \cdot i$, $L_{\mathrm{CSP}(N)}(S_i) = 3 \cdot i + 1$ and $L_{\mathrm{CSP}(N)}(K) = 2 \cdot n$. This numbering corresponds the the ordering $(X_0, S_0, X_1, S_1, X_2, S_2, X_3, S_3, K)$. We have $\mathrm{c}(L_{\mathrm{CSP}(N)})(P) = 5$ .*

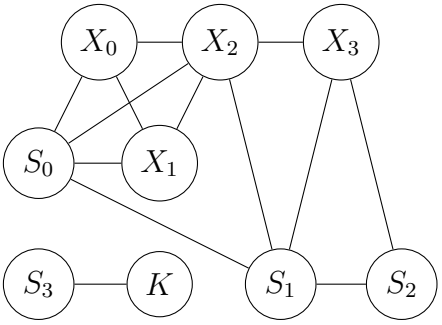



Figure 1.13: Primal graph of CSP(N)

**Theorem 10.** *Given a Petri-net $N$ with $n$ places with a numbering $L_N$ and an integer $K$, there exists an algorithm to find a siphon of size $K$ if such a siphon exists in $N$ in $\mathbf{O}(|N|^{\mathrm{c}(L_N)+3} \cdot \log |N|)$.*

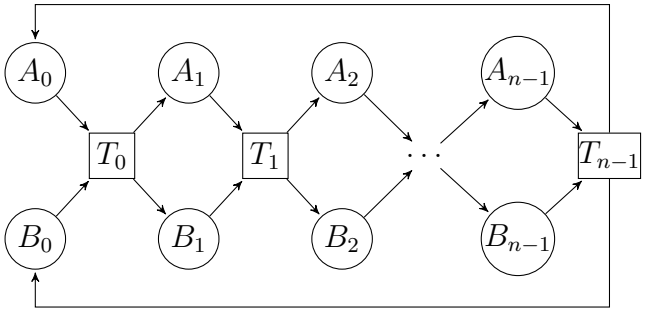
*Proof.* By lemma 2, $\mathrm{c}(\mathrm{CSP}(L_N)) = \mathrm{c}(L_N) + 2$ Therefore, $\mathrm{tw}(\mathrm{CSP}(L_N)) \leq \mathrm{c}(L_N) + 2$ by lemma 8. Moreover, $|N| = \mathbf{O}(|P|)$. Theorem 9 concludes. □

**Theorem 11.** *Given a Petri net $N$ with $n$ places with a numbering $L_N$, and a subset of places $Q$, there exists an algorithm to find a minimal siphon containing $Q$ in $N$ in $\mathbf{O}(|N|^{\mathrm{c}(L_N)+5} \cdot (\log |N|)^2)$.*

*Proof.* The constraint satisfaction problem $\mathrm{CSP}'$ obtained from $\mathrm{CSP}(N)$ by adding the constraints $X_i = 1$ for all $p_i \in Q$ has the same cut-width as $\mathrm{CSP}(N)$. Therefore, a siphon $S$ containing $Q$ with minimal cardinality can be found by iterating the resolution of $\mathrm{CSP}'$ by a dichotomic search among the values of $K$ between 1 and $n$. If $S$ is minimal (which can be checked polynomially in $\mathbf{O}(|N|^2)$), then this is a minimal siphon containing $Q$ in $N$. If $S$ is not minimal, then there is no minimal siphon containing $Q$ in $N$. □

**Proposition 9.** *The count of minimal siphons in Petri nets is not polynomially bounded by cut-width.*

*Proof.* Given an integer $n \geq 2$, consider the Petri-net $N$ depicted below, which has $2^n$ minimal siphons.




The cut-width of $N$ is 5. Indeed, the primal graph of the underlying hyper-graph of $N$ is $G = (V, E)$ such that $V = \{A_i, B_i : 0 \leq i < n\}$ and $E = \bigcup_{0 \leq i < n} \{A_i, B_i\} \times \{A_{i+1 \bmod n}, B_{i+1 \bmod n}\}$. $G$ is the non-oriented underlying graph. We have $\mathrm{c}(G) \leq 6$. Indeed, the numbering $L_G$ such that for all $0 \leq i < n$, $L_G(A_i) = 2 \cdot i$, $L_G(B_i) = 2 \cdot i + 1$ is such that $\mathrm{c}(L_G) = 4$. □

### 1.3.2 Linear Time Complexity Result

We show that, given a parameter $k > 0$, Rec-MESP can be decided in linear time for Petri nets with tree-width bounded by $k$. We prove also that given a parameter $k > 0$, deciding the Siphon Trap Property can be done in linear time for Petri nets with tree-width bounded by $k$. These results follow from Courcelle's theorem which states that every graph property definable in monadic second-order logic can be decided in linear time on graphs of bounded tree-width. The monadic second-order logic is the extension of the first-order logic that allows quantification over monadic unary predicates (i.e.: sets). Thus, non-unary predicates, as well as functions, may

appear in monadic second-order languages, but they may not be quantified over. Hence, monadic Second Order logics formulas are built up from atomic formulas using the usual boolean connectives ($\vee$ ; $\wedge$; $\neg$ ; $\rightarrow$ ; $\leftrightarrow$), quantification over individual variables and quantification over set variables. It is shown that monadic second-order logic, where quantifications over sets of vertices and sets of edges are used, is a reasonably powerful logical language for which one can obtain decidability results.

Our main theorem states the linear time complexity of the Rec-MESP in Petri nets with bounded tree-width.

**Theorem 12.** *For $k > 0$, there exists an algorithm to find a minimal siphon containing a subset of places $Q$ in linear time for Petri nets of tree-width bounded by $k$.*

*Proof.* Our theorems are corollaries of the Courcelle's theorem that declares the following:

**Theorem 13.** *(Courcelle's theorem)[24] If a class $C$ of graphs is definable in monadic second-order logic (MSO) then for any fixed $k > 0$, given a $k$-width tree decomposition of a graph $G$, there exists a linear-time algorithm which recognizes if $G \in C$.*

Given a Petri-net with $n$ places and $m$ transitions, the incidence between vertices (places and transitions) and edges is represented by a binary relation *edge*. To separate places from transitions, we introduce the unary predicate *place*. A siphon is a set of places whose predecessors are also successors.

The fact that a set of places $S$ is a siphon can be written as the following logical expression:

$$\textsc{siphon}(\mathrm{S}) : \forall v(v \in S \Rightarrow place(v)) \wedge \exists v(v \in S) \wedge$$
$$\forall t(\exists v(v \in S \wedge edge(t, v)) \Rightarrow \exists v(v \in S \wedge edge(v, t)))$$

$S$ is minimal when $\textsc{Min-Siphon}(\mathrm{S})$ holds where:

$$\textsc{Min-Siphon}(\mathrm{S}) : \textsc{siphon}(\mathrm{S}) \wedge \forall S'(\textsc{siphon}(\mathrm{S'}) \wedge$$
$$\forall v(v \in S' \Rightarrow v \in S) \Rightarrow \forall v(v \in S \Rightarrow v \in S'))$$

$S$ contains the given set of places $Q$ is trivially written as:

$$\forall v(v \in Q \Rightarrow v \in S)$$

Hence, the existence of a minimal siphon $S$ containing a given set of places $Q$ is represented by the following monadic second order logic expression:

$$\textsc{Min-Siphon-containing}(\mathrm{Q}) : \exists S(\textsc{siphon}(\mathrm{S}) \wedge \forall S'(\textsc{siphon}(\mathrm{S'}) \wedge$$
$$\forall v(v \in S' \Rightarrow v \in S) \Rightarrow \forall v(v \in S \Rightarrow v \in S')) \wedge \forall v(v \in Q \Rightarrow v \in S))$$

□

**Theorem 14.** *For $k > 0$, the Siphon Trap Property (STP) can be decided in linear time for Petri nets of tree-width bounded by $k$.*

*Proof.* We define a unary predicate *marked* over the set of places. The STP holds when every siphon contains a marked trap which can be written formally by the following expression:

$$\text{STP} : \forall S(\textsc{siphon}(\text{S}) \Rightarrow \exists S'(\forall v(v \in S' \Rightarrow v \in S) \wedge$$

$$\textsc{trap}(\text{S'}) \wedge \exists v(v \in S' \wedge marked(v))))$$

A trap is the dual notion of siphon (a set of places whose successors are also predecessors). The unary predicate corresponding to a trap is:

$$\textsc{trap}(\text{S}) : \forall v(v \in S \Rightarrow place(S)) \wedge \exists v(v \in S) \wedge$$

$$\forall t(\exists v(v \in S \wedge edge(v,t)) \Rightarrow \exists v(v \in S \wedge edge(t,v)))$$

□

## 1.4 Petri Net Structures and CTL Properties

### 1.4.1 Infinite State Computation Tree Logic

Temporal logics have been introduced by Amir Pnueli in [84] as a mean to reason about the dynamic of concurrent programs. Temporal logics may differ according to how they handle branching in the underlying computation tree. In a linear temporal logic, operators are provided for describing events along a single computation path. In a branching-time logic the temporal operators quantify over the paths that are possible from a given state.

The computation tree logic CTL$^*$ combines both branching-time and linear-time operators. In this logic a path quantifier can prefix an assertion composed of arbitrary combinations of the usual linear-time operators. The temporal operators are the following where $p$ and $q$ are atomic propositions.

1. Path quantifier:
   - **A** for every path
   - **E** there exists a path
2. Linear-time operators:
   - **X**$p$ $p$ holds next time.
   - **F**$p$ $p$ holds sometime in the future
   - **G**$p$ $p$ holds globally in the future

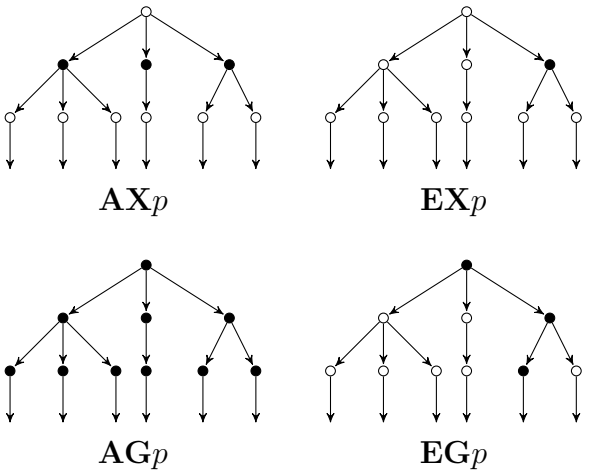



Figure 1.14: Reachability trees for **AX**$p$, **EX**$p$, **AG**$p$, and **EG**$p$.

- $p$**U**$q$ $p$ holds until $q$ holds

CTL [18] is a restricted subset of CTL$^*$ [18] where each of the linear-time operators **X**, **F**, **G** and **U** must be immediately preceded by a path quantifier (e.g. **AG**(**EF**$p$)). Figure 1.14 shows a graphical interpretation of some temporal operators of CTL, namely the operator **AX**, **EX**, **AG**, and **EG** that deal with properties of interest in the following. The root of the computation tree is the initial state of the reachability graph. States represented by white nodes are such that the formula $p$ does not hold whereas $p$ holds in states represented by black nodes.

In the following, we present the syntax and the semantics of CTL$^*$.

**Definition 38.** *(Syntax of* CTL$^*$*) Let $p$ be an atomic proposition. Formulas in* CTL$^*$ *are either state-formulas or path-formulas which satisfy the following rules:*

1. *$p$ is a state-formula*
2. *If $\Phi$ is a state-formula, then $\neg\Phi$ is a state-formula*
3. *If $\Phi$ and $\Psi$ are state-formulas, then $\Phi \vee \Psi$ is a state-formula*
4. *If $\varphi$ is a path-formula, then $E\varphi$ and $A\varphi$ are state-formulas*
5. *Anything else is not a state-formula.*

*Path-formulas satisfy the following rules:*

1. *If $\Phi$ is a state-formula, then $X\Phi$ is a path-formula*
2. *If $\Phi$ and $\Psi$ are state-formulas, then $\Phi U \Psi$ is a path-formula*
3. *Anything else is not a path-formula.*

Intuitively, a state-formula express a property of a state, while a path-formula express a property of a path, i.e., an infinite sequence of states.

The interpretation of CTL$^*$ is defined in terms of a Kripke structure. We recall the following definition and notations:

**Definition 39.** *(Kripke structure) A Kripke structure $\mathcal{S}$ is a tuple $\mathcal{S} = (S, \rightarrow, L, I)$ where*

- *$S$ is a set of states.*
- *$\rightarrow \subseteq S \times S$ is a total relation (i.e. for any state $s \in S$, there exists a state $s'$ such that $s \rightarrow s'$).*
- *$L : S \rightarrow 2^{AP}$ is a labelling function over the set of atomic propositions $AP$. It associates to each state, the set of atomic propositions true in that state.*
- *$I$ is the set of initial states.*

**Example 13.** *Let $AP = \{p, q\}$ be the set of atomic propositions. $p$ and $q$ can model arbitrary boolean properties of the system that the Kripke structure is modelling. The figure 1.15 illustrates a Kripke structure $M = (S, \rightarrow, L, I)$, where*

- $S = \{s1, s2, s3\}$.
- $\rightarrow = \{(s1, s2), (s2, s1)(s2, s3), (s3, s3)\}$.
- $L = \{(s1, p, q), (s2, q), (s3, p)\}$.
- $I = \{s1\}$.

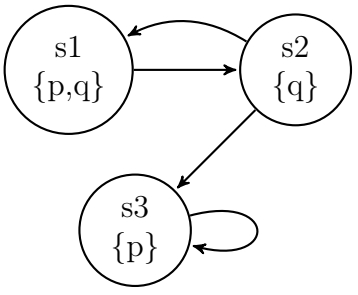



Figure 1.15: An example of Kripke structure

It is worth noticing that we do not restrict $S$ to be finite in the definition. Indeed, we consider infinite Kripke structures in this section, and the next section defines an abstraction framework for reducing the infinite sets of states into finite sets while preserving the properties presented in this section.

**Definition 40.** *(Path) A path in Kripke structure $M = (S, \rightarrow, L, I)$ is an infinite sequence of states $s_0 s_1 s_2 ...$ such that $(s_i, s_{i+1}) \in \rightarrow$ for all $i > 0$.*

A path is thus an infinite sequence of states such that between successive states transitions do exist. For path $\sigma = s_0 s_1 s_2 ...$ and integer $i > 0$ we use $\sigma[i]$ to denote the $(i+1)^{th}$ state of $\sigma$ , i.e., $\sigma[i] = s_i$. The set of paths that start in state $s$ is denoted $Paths(s)$. As each state in a Kripke structure is required to have at least one successor, it follows $Paths(s) \neq \emptyset$ for any state $s$.

For any Kripke structure $M = (S, \rightarrow, L, I)$ and state $s \in S$ there is an infinite computation tree with root labelled $s$ such that $(s_i, s_j)$ is an arc in the tree if and only if $(s_i, s_j) \in \rightarrow$. This tree is obtained by unfolding the Kripke structure at state $s$.

**Definition 41.** *(Semantics of* CTL$^*$*) Let $p \in AP$ be an atomic proposition, $M = (S, R, L, I)$ be a Kripke structure, $s \in S$, $\Phi$, $\Psi$ be* CTL$^*$ *state-formulas, and $\varphi$ be a* CTL$^*$ *path-formula. The satisfaction relation $\models$ is defined for state-formulas by:*

- $s \models p$ *iff* $p \in L(s)$
- $s \models \neg\Phi$ *iff* $\neg(s \models \Phi)$
- $s \models \Phi \vee \Psi$ *iff* $(s \models \Phi) \vee (s \models \Psi)$
- $s \models \mathbf{E}\varphi$ *iff* $\exists \sigma \in Paths(s).\ \sigma \models \varphi$
- $s \models \mathbf{A}\varphi$ *iff* $\forall \sigma \in Paths(s).\ \sigma \models \varphi$

*For path $\sigma$ the satisfaction relation $\models$ for path-formulas is defined by:*

- $\sigma \models \mathbf{X}\Phi$ *iff* $\sigma[1] \models \Phi$
- $\sigma \models \Phi \mathbf{U} \Psi$ *iff* $\exists j \geq 0. (\sigma[j] \models \Psi \wedge (\forall 0 \geq k < j. \sigma[k] \models \Phi))$

Since Petri nets may have unbounded markings however, the set of states is infinite and standard model-checking techniques can not be directly applied. In this section we study the theoretical links between CTL and siphons and traps in Petri nets, and in the next section we study a boolean abstraction which makes these links effective to speed up the verification of some CTL properties.

### Infinite Kripke Structures for Petri Nets

In the following definition, we associate an infinite Kripke structure to a general Petri net. We describe a total transition relation by adding loops around deadlock markings (i.e. markings from which no transition can be fired).

**Definition 42.** *For any Petri net $PN = (P, T, W, m_0)$, the infinite Kripke structure induced by $PN$ is the structure $\mathcal{S}^{PN} = (\mathbb{N}^P, \to, L, \{m_0\})$ where for all $m, m' \in \mathbb{N}^P$, $m \to m'$ if either there exists a transition $t \in T$ such that $m \xrightarrow{t} m'$, or $m = m'$ and there doesn't exist any transition $t \in T$ and marking $m'' \in \mathbb{N}^P$ such that $m \xrightarrow{t} m''$.*

Note that there are other available choices of the completion of the transition relation such as adding loops around all markings, or the introduction of fresh states $\bot_S$ for every $S \subseteq AP$, *etc.*

### Traps, siphons and CTL

Given a Kripke structure associated to a Petri net, we provide the relationship between structural properties of Petri nets, traps and siphons, and their characterisation in CTL.

**Definition 43.** *A subset of markings $M \subseteq \mathbb{N}^P$ is invariant with respect to a Petri net $PN$ if for all $m \in M$, for all transition $t$ and marking $m' \in \mathbb{N}^P$ such that $m \xrightarrow{t} m'$, we have $m' \in M$.*

**Theorem 15.** *Let $\phi$ be a* CTL *formula and $M_\phi = \{m \in \mathbb{N}^p \mid (\mathcal{S}^{PN}, m) \models \phi\}$. $M_\phi$ is invariant with respect to $PN$ if and only if for all $m \in \mathbb{N}^P$, $(\mathcal{S}^{PN}, m) \models \phi \Rightarrow \mathbf{AX}\phi$*

*Proof.* Suppose that $M_\phi$ is invariant with respect to $PN$. Then, for all $m \in \mathbb{N}^P$, we show that $(\mathcal{S}^{PN}, m) \models \phi \Rightarrow \mathbf{AX}\phi$: indeed, if $(\mathcal{S}^{PN}, m) \models \phi$, then $m \in M_\phi$, therefore, for all $m' \in \mathbb{N}^P$ such that $m \to m'$, there exists a transition $t$ such that $m \xrightarrow{t} m'$, therefore, since $M_\phi$ is invariant with respect to $PN$, $m' \in M_\phi$, thus $(\mathcal{S}^{PN}, m') \models \phi$ and then $(\mathcal{S}^{PN}, m) \models \mathbf{AX}\phi$.

Conversely, suppose that for all $m \in \mathbb{N}^P$, $(\mathcal{S}^{PN}, m) \models \phi \Rightarrow \mathbf{AX}\phi$. Then, for all $m \in \mathbb{N}^P$, if $m \in M_\phi$, we show that for all $m' \in \mathbb{N}^P$ such that there exists a transition $t$ such that $m \xrightarrow{t} m'$, we have $m' \in M_\phi$. Indeed, since $m \in M_\phi$, we have $(\mathcal{S}^{PN}, m) \models \phi$, therefore $(\mathcal{S}^{PN}, m) \models \mathbf{AX}\phi$. Since there exists a transition $t$ such that $m \xrightarrow{t} m'$, we have $m \to m'$, therefore $(\mathcal{S}^{PN}, m') \models \phi$, thus $m' \in M_\phi$. □

It is worth noticing that $\phi \Rightarrow \mathbf{AX}\phi$ is logically equivalent to $\phi \Rightarrow \mathbf{AG}\phi$.

For any subset $P' \subseteq P$ of places, let $\phi_{P'} = \bigvee_{p \in P'} m_p \geq 1$

**Corollary 1.** *The set $P' \subseteq P$ is a trap in $PN$ if and only if for all $s \in \mathbb{N}^P$, $(\mathcal{S}^{PN}, s) \models \phi_{P'} \Rightarrow \mathbf{AG}\phi_{P'}$.*

*Proof.* According to proposition 3, $P'$ is a trap if and only if $M_{\phi_{P'}}$ is invariant with respect to $PN$. □

**Corollary 2.** *The set $P' \subseteq P$ is a siphon in $PN$ if and only if for all $s \in \mathbb{N}^P$, $(\mathcal{S}^{PN}, s) \models \neg\phi_{P'} \Rightarrow \mathbf{AG}\neg\phi_{P'}$.*

*Proof.* According to proposition 4, $P'$ is a siphon if and only if $M_{\neg\phi_{P'}}$ is invariant with respect to $PN$. □

### 1.4.2 Boolean Abstractions, Boundedness Conditions and Boolean CTL Model-Checking

Model-checking methods [18] are powerful techniques for verifying temporal logic properties of systems modelled by finite state machines. Although the set of states has to be finite by hypothesis, the number of states can be very large and one main challenge of model-checking methods is to deal with the state space explosion problem. State-of-the-art symbolic model-checkers based on binary decision diagrams, such as for instance NuSMV [11] for proving Boolean CTL properties, have shown their capability to verify extremely large systems in circuit design and program verification. This is also the case in biochemical networks where Boolean CTL formulas allow us to express a broad spectrum of biological properties such as accessibility, fixed points, stability or oscillations, and verify them by symbolic model-checking [14]. Key to the application of these methods is the reduction of an infinite state machine such as a Petri net to a finite state machine. In this section, we formalize the reduction of Petri nets by quotients and the boolean abstraction of a Petri net, and study the CTL properties preserved by these reductions.

#### Quotient Kripke Structures

Let $\mathcal{S} = (S, \to, L, I)$ be a (potentially infinite) Kripke structure, the definition of quotient Kripke structure is natural as it is done in [3]. Here, $\equiv$ is an arbitrary equivalence relation over $S$. States of $S$ are written $s$, $s'$, .... Equivalence classes of $S/\equiv$ are written $c$, $c'$, .... The equivalence class of a state $s \in S$ is written $[s]$.

**Definition 44.** *The quotient Kripke structure of $\mathcal{S}$ by $\equiv$ is*

$$(\mathcal{S}/\equiv) = (S/\equiv, \to_{\equiv}, [L], [I])$$

*where*

- $\to_{\equiv} \in S/\equiv \times S/\equiv$ *is the relation such that* $c \to_{\equiv} c'$ *when there exist* $s \in c$ *and* $s' \in c'$ *with* $s \to s'$,
- $[L] : c \in S/\equiv \mapsto \bigcup_{s \in c} L(s)$,
- $[I] = \{c \in S/\equiv \mid \exists s \in I, s \in c\}$

This quotient Kripke structure is similar to the quotient transition system introduced by Alur and al. in [3] to find classes of infinite transition systems whose analysis can be performed on equivalent but finite transition systems. The difference between the two quotients is that Alur and al. restrict the equivalence relation to be proposition-preserving (an equivalent relation is said to be proposition-preserving if for all states $p, q \in Q$ and all propositions $\pi$, if $p \equiv q$ and $p \models \pi$ then $q \models \pi$) while we consider any equivalence relation.

**Proposition 10.** *$\mathcal{S}/\equiv$ is a Kripke structure.*

*Proof.* It amounts to show that $\rightarrow_{\equiv}$ is total: for all $c \in S/\equiv$, there exists at least one $s \in c$, moreover $\rightarrow$ is total, therefore there exists $s' \in S$ such that $s \rightarrow s'$, thus $c \rightarrow_{\equiv} [s']$. □

A common usage of quotients is the thresholding of states that allows us to reduce an infinite number of states to a finite number with some fixed maximum value.

**Definition 45.** *For any Petri net PN and for all $k \in \mathbb{N}$, the thresholding Kripke structure $\mathcal{S}^{PN}_{\leq k}$ is $\mathcal{S}^{PN}_{\leq k} = \mathcal{S}^{PN}/\equiv_{\leq k}$ where $\equiv_{\leq k}$ is such that $s \equiv_{\leq k} s'$ if and only if for all $p$ with $s_p \neq s'_p$, we have $s_p \geq k$ and $s'_p \geq k$.*

$\equiv_{\leq k}$ can also be written as $s \equiv_{\leq k} s'$ if and only if for all $p$, $min(s_p, k) = min(s'_p, k)$. $\mathcal{S}^{PN}_{\leq k}$ is a finite Kripke structure which faithfully captures transitions between markings with at most $k$ tokens in each place, and collapses all markings which differ only on places with more than $k$ tokens.

### Preservation of Boolean CTL Properties

We are interested in CTL properties which are preserved by quotient of the Kripke structure.

**Definition 46.** *A formula $\phi \in$ CTL is* strongly preserved *by $\equiv$ if for all $s \in S$, $(\mathcal{S}, s) \models \phi$ if and only if $(\mathcal{S}/\equiv, [s]) \models \phi$.*

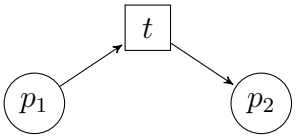
It is already known that bisimulation preserves CTL properties, a bisimilation is a special equivalence relation with the following property.

**Definition 47.** *(Bisimulation) Let $T = (Q, \rightarrow, L, I)$ be a Kripke structure. A proposition-preserving equivalence relation $\sim_B$ on $Q$ is a bisimulation of $T$ if for all states $p, q \in Q$, if $p \sim_B q$, then for all states $p' \in Q$, if $p \rightarrow p'$ then there exists a state $q' \in Q$ such that $q \rightarrow q'$ and $p' \sim_B q$'.*

**Theorem 16.** *[8] Let $T$ be a Kripke structure and let $\sim_B$ be a bisimulation of $T$. Then $T$ satisfies the* CTL *formula $\phi$ if and only if the bisimulation quotient $T/\sim_B$ satisfies $\phi$.*

However, our thresholding equivalence relation is not a bisimulation. An intuitive example is the following.

**Example 14.** *Consider the following Petri net:*




*Let $m = (2, 0)$, $m \equiv_{\leq 1} m' = (1, 0)$. We have $m \xrightarrow{t} m_1 = (1, 1)$ and there does not exist any marking $m_2$ such that $m' \rightarrow m_2$ and $m_1 \equiv_{\leq 1} m_2$: the only reachable marking from $m'$ is $(0, 1)$ and we have $(0, 1) \not\equiv_{\leq 1} (1, 1)$. Hence $\equiv_{\leq 1}$ is not a bisimulation.*

We are not interested in relations preserving all CTL properties but, on the contrary, in characterizing a family of CTL properties preserved by a a given equivalence relation.

**Definition 48.** *A formula $\phi \in$ CTL is* weakly preserved *by $\equiv$ if (for all $s \in S$, $(\mathcal{S}, s) \models \phi$) if and only if (for all $c \in S/\equiv$, $(\mathcal{S}/\equiv, c) \models \phi$)).*

The second form of preservation is not common, it is a special form of preservation that states that a formulas is verified by all states $s \in S$ when it is verified by all classes $c \in S/\equiv$. We introduce this latter form of preservation to deal with quantifier operators as it is shown in Theorem 18. Obviously, strong preservation subsumes weak preservation.

**Proposition 11.** *For the equivalence relation $\equiv_{\leq k}$, the atomic propositions $m_p \leq n$ and $m_p = n$ are strongly preserved when $n < k$, and the atomic propositions $m_p \geq n$ are strongly preserved when $n \leq k$.*

*Proof.* $\equiv_{\leq k}$ can also be written as $s \equiv_{\leq k} s'$ if and only if $min(s, k) = min(s', k)$. We want to prove that: $\forall k$, $\forall n$, $n < k \Leftrightarrow \forall m$, $(m \leq n \Leftrightarrow \forall m' s.t. m' \equiv_{\leq k} m, m' \leq n)$. $n < k$ and $m \leq n$, then $min(m', k) = m' = m$ and thus $m' \leq n$. Now, $n < k$ and $m' \leq n$, then $min(m, k) = m' = m$ and thus $m \leq n$. Conversely, we have for all $m$, $m \leq n$ is equivalent to for all $m'$ such that $min(m, k) = min(m', k)$, we have $m' \leq n$. Supposing that $k \leq n$ will give us contradiction with $m = n + 1$, hence $n < k$. Similarly, for all $n$ and $k$, $n < k$ if and only if for all $m$, $m = n$ is equivalent to $\min(m, k) = n$. Finally, for all $n$ and $k$, $n \leq k$ if and only if for all $m$, $m \geq n$ is equivalent to for all $m'$ such that $\min(m, k) = min(m', k)$, we have $m' \geq n$. □

In general, the preserved formulas highly depend on $\equiv$: if $\equiv$ is the identity relation, then all formulas are preserved, and if $\equiv$ collapses all states to a single equivalence class, only tautologies and self-contradictions are preserved.

**Theorem 17.** *The class of strongly preserved formulas is closed by the Boolean connectives,* i.e.*, for all Kripke structures $\mathcal{S}$ and equivalence relation $\equiv$, if $\phi$ and $\psi$ are formulas strongly preserved by $\equiv$, then $\phi \wedge \psi$, $\phi \vee \psi$ and $\neg\phi$ are strongly preserved by $\equiv$.*

*Proof.* Let $\phi$ and $\psi$ be strongly preserved formulas. For all state $s$, $(\mathcal{S}, s) \models \phi \wedge \psi$, if and only if $(\mathcal{S}, s) \models \phi$ and $(\mathcal{S}, s) \models \psi$, if and only if $(\mathcal{S}/\equiv, [s]) \models \phi$ and $(\mathcal{S}/\equiv, [s]) \models \psi$, if and only if $(\mathcal{S}/\equiv, [s]) \models \phi \wedge \psi$. Similarly, for all state $s$, $(\mathcal{S}, s) \models \phi \vee \psi$, if and only if $(\mathcal{S}, s) \models \phi$ or $(\mathcal{S}, s) \models \psi$, if and only if $(\mathcal{S}/\equiv, [s]) \models \phi$ or $(\mathcal{S}/\equiv, [s]) \models \psi$, if and only if $(\mathcal{S}/\equiv, [s]) \models \phi \vee \psi$, Finally, for all state $s$, $(\mathcal{S}, s) \models \neg\phi$, if and only if $(\mathcal{S}, s) \not\models \phi$, if and only if $(\mathcal{S}/\equiv, [s]) \not\models \phi$, if and only if $(\mathcal{S}/\equiv, [s]) \models \neg\phi$. □

In general, the class of strongly preserved formulas is not closed by temporal logic operators. For example, consider the structure $\mathcal{S} = (\{0, 1, 2, 3\}, \{0 \to 1, 1 \to 1, 2 \to 3, 3 \to 3\}, (0, 1, 2 \mapsto \emptyset; 3 \mapsto \{\mathrm{a}\}))$, $\{0\}$ and the equivalence relation $\equiv$ induced by the equation system $\{0 \equiv 2\}$. The formula $\phi = \mathrm{a}$ is strongly preserved by $\equiv$, but not $\mathbf{AX}\phi$.

**Theorem 18.** *The class of weakly preserved formulas is closed by conjunction and disjunction,* i.e.*, for all Kripke structures $\mathcal{S}$ and equivalence relation $\equiv$, if $\phi$ and $\psi$ are formulas weakly preserved by $\equiv$, then $\phi \wedge \psi$ and $\phi \vee \psi$ are weakly preserved by $\equiv$. Moreover, if $\phi$ is a strongly preserved formula, then $\mathbf{AX}\phi$ is a weakly preserved formula.*

*Proof.* Let $\phi$ and $\psi$ be weakly preserved formulas. For all state $s$, $(\mathcal{S}, s) \models \phi \wedge \psi$, if and only if for all state $s$, $(\mathcal{S}, s) \models \phi$ and $(\mathcal{S}, s) \models \psi$, if and only if for all state $s$, $(\mathcal{S}/\equiv, [s]) \models \phi$ and $(\mathcal{S}/\equiv, [s]) \models \psi$, if and only if for all state $s$, $(\mathcal{S}/\equiv, [s]) \models \phi \wedge \psi$. Similarly, for all state $s$, $(\mathcal{S}, s) \models \phi \vee \psi$, if and only if $(\mathcal{S}, s) \models \phi$ or $(\mathcal{S}, s) \models \psi$, if and only if $(\mathcal{S}/\equiv, [s]) \models \phi$ or $(\mathcal{S}/\equiv, [s]) \models \psi$, if and only if $(\mathcal{S}/\equiv, [s]) \models \phi \vee \psi$.

Let $\phi$ be a strongly preserved formula. Suppose that for all $s \in S$, $(\mathcal{S}, s) \models \mathbf{AX}\phi$. For all $c \in S/\equiv$, we show that $(\mathcal{S}/\equiv, c) \models \mathbf{AX}\phi$, which amounts to proving that, for all $c' \in S/\equiv$ such that $c \rightarrow_{\equiv} c'$, we have $(\mathcal{S}/\equiv, c') \models \phi$. Since $c \rightarrow_{\equiv} c'$, then there exists $s \in c$ and $s' \in c'$ such that $s \rightarrow s'$. Since $(\mathcal{S}, s) \models \mathbf{AX}\phi$ and $s \rightarrow s'$, we have $(\mathcal{S}, s') \models \phi$, then $(\mathcal{S}, [s']) \models \phi$, that is to say $(\mathcal{S}/\equiv, c') \models \phi$. Conversely, suppose that for all $c \in S/\equiv$, $(\mathcal{S}/\equiv, c) \models \mathbf{AX}\phi$. For all $s \in S$, we show that $(\mathcal{S}, s) \models \mathbf{AX}\phi$, which amounts to proving that, for all $s' \in S$ such that $s \rightarrow s'$, we have $(\mathcal{S}, s') \models \phi$. Since $s \rightarrow_{\equiv} s'$, then $[s] \rightarrow_{\equiv} [s']$. Since $(\mathcal{S}/\equiv, [s]) \models \mathbf{AX}\phi$ and $[s] \rightarrow_{\equiv} [s']$, we have $(\mathcal{S}/\equiv, [s']) \models \phi$, then $(\mathcal{S}/\equiv, s') \models \phi$. □

**Corollary 3.** *The set $P' \subseteq P$ is a trap in $PN$ if and only if for all $[s] \in \{\{0\}, \mathbb{N}_{\geq 1}\}^P$, $(\mathcal{S}^{PN}_{\leq 1}, [s]) \models \phi_{P'} \Rightarrow \mathbf{AG}\phi_{P'}$.*

*Proof.* From theorem 17 and proposition 11, it follows that $\phi_{P'}$ is strongly preserved by $\equiv_{\leq k}$. Then, according to theorem 18, $\phi_{P'} \Rightarrow \mathbf{AG}\phi_{P'}$ is weakly preserved, and is a characterization of traps according to corollary 1. □

**Corollary 4.** *The set $P' \subseteq P$ is a siphon in $PN$ if and only if for all $[s] \in \{\{0\}, \mathbb{N}_{\geq 1}\}^P$, $(\mathcal{S}^{PN}_{\leq 1}, [s]) \models \neg\phi_{P'} \Rightarrow \mathbf{AG}\neg\phi_{P'}$.*

*Proof.* Similarly, $\neg\phi_{P'} \Rightarrow \mathbf{AG}\neg\phi_{P'}$ is weakly preserved, and is a characterization of siphons according to corollary 2. □

### Evaluation

Evaluation is carried on in the Biochemical Abstract Machine (Biocham)[1] [9] which is a modelling environment for systems biology, with some unique features for static analysis or for inferring unknown model parameters from temporal logic constraints. Biocham is mainly composed of :

- a rule-based language for modelling biochemical systems (compatible with SBML),

[1] http://contraintes.inria.fr/Biocham/

- several simulators (boolean, differential, stochastic),

- a temporal logic based language to formalize the temporal properties of a biological system and validate models with respect to such specifications,

- unique features for developing/correcting/completing/coupling models, including the inference of kinetic parameters in high dimension from temporal logic constraints.

A Biocham model is a set of reaction rules given with an initial state. The formal semantics of a Biocham model is a Kripke structure, that is a triple formed of a set of states, a transition relation between states and a labelling function associating to each state the set of atomic propositions true in that state. One can associate to a Biocham model a Kripke structure, where the set of states S is the set of all tuples of boolean values denoting the presence or absence of the different biochemical compounds (molecules, genes and abstract processes), the transition relation R is the union (i.e. disjunction) of the relations associated to the reaction rules, and the labelling function L simply associates to a given state the set of biochemical compounds which are present in the state. Reaction rules in Biocham are asynchronous in the sense that one reaction rule is fired at a time (interleaving semantics), hence the transition relation is the union of the relations associated to the reaction rules.

In section 1.4.2, we assigned to a Petri net underlying a biochemical reaction network a Kripke structure, and we have developed links between the structural properties of siphons and traps in Petri nets and their dynamical properties in Computation Tree Logic. For this, we have formally defined the infinite Kripke structure associated to a Petri net, and their quotients by equivalence relations which constitute formal abstractions in the sense of Galois connections. Since Biocham is developed in the context of Kripke structure, it is convenient to carry out the evaluation of our approach in the context of Biocham. To evaluate how much can be gained, a simple program based on GNU-Prolog solver was implemented to check whether a set of places is a siphon or a trap and a comparison between its performance and the state-of-the-art symbolic model-checker NuSMV [11] used in Biocham is hold on some examples from systems biology.

**Example 15.** *Let us first consider the small model of the bacteriophage T7 [2] which consists of* 6 *simple reactions:*

```
gen => tem.
tem => _.
tem => tem + gen.
gen + struc => virus.
tem => tem + struc.
struc => _.
```

The system has one minimal siphon: $\{gen, tem\}$. The corresponding CTL queries are the following[2]:

```
check_ctl(Ai((!(gen) & !(tem)) -> AX((!(gen) & !(tem))))).
```

The state-of-the-art symbolic model-checker NuSMV [11] used in Biocham [9] returns a positive answer in less than five milliseconds.

For some models however, especially with those models containing many parallel pathways, symbolic model-checkers may fail to verify simple CTL formulae in reasonable time. This is the case, for instance, of Schoeberl's model of the MAP kinase cascade activated by surface and internalized EGF receptors [92], numbered model 19 in the Biomodels.net repository [3]. This model contains 100 places and 242 transitions. Using our constraint satisfaction algorithm, the sets of 13 minimal siphons and 15 minimal traps are computed in less than one second (almost 20 milliseconds) [4]. In this model, {EGFRideg} is a trap. Intuitively, this means that, if {EGFRideg} is present at a certain state, it will never become absent. Veryfing if {EGFRideg} is a trap using our program is very fast and can be done in less than five milliseconds. In contrast, NuSMV fails to verify the corresponding CTL formulae in a reasonable time: NuSMV does not provide any result within a timeout of ten minutes. This property can be written, using Biocham notation, as:

```
check_ctl(Ai(EGFRideg -> AX(EGFRideg))).
```

In the same manner, species phosphatase3, ERK−PP−P'ase3, ERK−P−P'ase3, ERKi−PP−P'ase3, ERKi−P−P'ase3, identified respectively by x60, x61, x62, x84 and x85, form a siphon and a trap in the same time.

This ascertainment is intuitively reasonable since these species correspond to different forms of the phosphatase. So their sum remains constant. The siphon corresponds to the following CTL formula:

```
check_ctl(Ai((!(x60)&!(x61)&!(x62)&!(x84)&!(x85))
  -> AX((!(x60)&!(x61))&!(x62)&!(x84)&!(x85))))
```

Here again NuSMV does not provide any result in less than ten minutes, whereas checking if {x60,x61,x62,x84,x85} is a siphon is instantaneous and is done in less than five milliseconds.

It is not surprising that simple structural Petri net properties are faster to check directly than using a symbolic model checker whose limiting factor is the state space computation step, which is performed by NuSMV via an ordinary Breath First Search (BFS) strategy. There are more advanced techniques such as saturation which may outperform BFS and which is supported by, e.g., the symbolic CTL model checking tools SMART and MARCIE.

[2] we use BIOCHAM syntax for CTL formulae and thus include the $Ai$ operator which expresses that the formula must be checked for all initial states.

[3] dated January 2011

[4] computation time on a PC with an intel Core2 Quad processor 2.8GHz and 8Go of memory.

A boolean model computing all minimal siphons (in term of inclusion) is presented in chapter 3. The program is tested on a benchmark of 403 models from Biomodels.net database. It is very efficient and scales up well even on large models.

# Chapter 2

# Petri Nets for Biochemical Networks

## Contents



## Introduction

Biochemical reaction systems are naturally bipartite (species and interactions), concurrent (interactions can occur independently and in parallel) and stochastic (the timing behaviour of the interactions is governed by stochastic laws).

Petri nets encompass the three characteristics aspects that distinguish biochemical networks: stochastic Petri nets have exactly these three characteristics.

However, dealing with stochastic models is difficult, the qualitative and the continuous abstractions are more popular.

## 2.1 Biological context

### 2.1.1 Systems Biology

Systems Biology can be defined as the study of all the elements in a biological system (genes, mRNAs, proteins, etc) and their relationships one to another in response to perturbations.

Systems biology involves:

- Collection of experimental data,
- Design of models,
- Testing and validation of these models.

An overview of Systems Biology is presented by Kitano, one of the pioneers in Systems Biology in [59]. Further references are available in [100, 37].

Models are built to answer specific questions about a biochemical system. They serve as an unambiguous representation of the acquired information and help to design new experiments to clarify our understanding.

### 2.1.2 Molecular Biology and Cellular Metabolism

A classical reference for some fundamental biological notions is the book *Molecular biology of the cell* [1]. Transcription and translation are the two steps required in the protein synthesis process. Proteins are molecules that work as a structural material, as enzymes, as antibodies, as transporters (hemoglobin), or as regulators of gene expression. The desoxyribonucleic acid (DNA) is a macro molecule present in all cells, it contains the genetic information about an organism. DNA forms a double helix in which two strands of DNA spiral about one other. There are four nucleotide in DNA: Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). The nucleotide A pairs only with T, whereas C pairs only with G. A gene is a portion of a continuous strand of DNA, from which a complex molecular machinery can read information and create a particular protein. The process of information transmission from DNA to proteins is called gene expression as shown in Figure 2.1.

Cellular metabolism includes complex sequences of controlled biochemical reactions. These processes permit organisms to grow and reproduce, maintain their structures and respond to environmental changes. The chemical reactions of

metabolism are organized into metabolic pathways, in which one chemical is transformed into another by a sequence of chemical reactions catalysed by enzymes. Enzymes are chemicals that speed up the rate of reaction between substances without themselves being consumed. The cell metabolism is then the sum of all chemical changes that take place in a cell through which energy and basic components are provided for vital processes, including the synthesis of new molecules and removal of others.

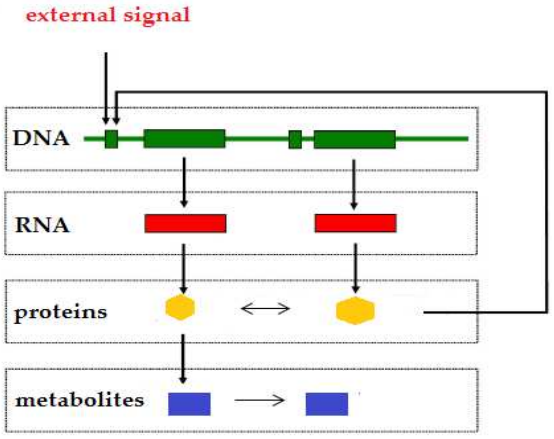



Figure 2.1: Information flow from genes to metabolites in cells

### 2.1.3 Biochemical Networks

Biochemical networks such as metabolic, regulatory, signalling or protein interaction networks can be viewed as interconnected processes forming a complex network of functional and physical interactions between chemical species. Modelling these networks is a way to have a global view of all the involved chemical reactions and can serve to suggest new interpretations or questions for experiment. Moreover, it can be the unique solution to analyse and extract information about cell metabolism specially when much substances are involved. Analysing these networks remains however a challenging problem in systems biology and in bioinformatics. In fact, biochemical networks have been under study for many decades. But the efforts were until recently limited to the determination of the components of the networks, rather than addressing structure of the interaction network.

To a better understanding of biochemical networks, we propose a brief introduction to the principal ones involved in cell metabolism in the following part.

- **Gene regulatory networks**

  A gene regulatory network is a set of genes, proteins, small molecules, and their mutual regulatory interactions. Two genes are connected if the expression of one gene modulates expression of another one by either activation or inhibition. Interactions between genes are not easy to model using Petri nets. In [98],

authors propose an approach to derive a standard Petri net model from a boolean regulatory network (where genes are ON or OFF). A case study is the Petri net modelling and analysis of the genetic regulatory network underlying nutritional stress.

- **Signal transduction networks**

  In biology, cell communication or signal transduction is the means by which cells respond to signals coming from outside. Signal transduction networks can be understood as gene regulation networks extended by signalling chains that contain different kinds of vertices and edges such as protein–protein interaction and phosphorylation. Processes referred to as signal transduction often engage a sequence of biochemical reactions inside the cell, which are carried out by enzymes and linked through second messengers. Such processes take place in a short time as a millisecond or as long as a few seconds. A signal transduction network can be graphically represented by a graph where nodes correspond to proteins and molecules whereas edges are reactions and processes (e.g. ligand/receptor binding protein conformational changes).

- **Protein interaction networks**

  Protein-protein interactions (PPI) are one of the most important components of biological networks. It is important to understand the structure and dynamics of PPIs in order to understand how the evolution of biological networks has contributed to diversification of the living organisms. They play a key role in determining the outcome of most cellular processes. These networks are modelled by graphs where vertices represent proteins and edges represent physical interactions between proteins. Figure 2.2 depicts a protein interaction network elaborated by Hawoong Jeong where red nodes correspond to essential protein yellow nodes correspond to growth- affecting protein and green nodes correspond to non-essential protein.

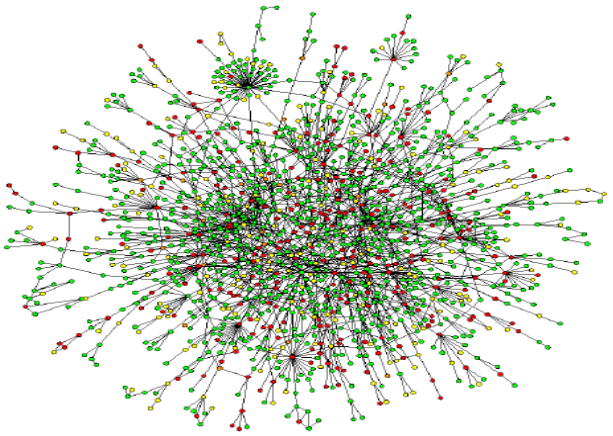


Figure 2.2: Map of yeast protein-protein interactions

- **Metabolic networks** A metabolic network is the biochemical modification of chemical compounds in living organisms and cells. This includes the biosynthesis of complex organic molecules (anabolism) and their breakdown (catabolism). A metabolic pathway is a connected sub-network of the metabolic network either representing specific processes or defined by functional boundaries. A metabolic pathway is a hyper-graph: the nodes represent the substances and the hyper-edges represent the reactions. A hyper-edge connects all substances of a reaction, and is directed from reactants to products and is labelled with the enzymes that catalyse the reaction. Hyper-graphs can be represented by bipartite graphs: Additionally to the nodes representing substances, the reactions are nodes and edges are binary relations connecting the substances of a reaction with the corresponding reaction node. This is a common modelling of metabolic pathways, e.g., for their simulation using Petri nets. Figure 2.3 illustrates the Petri net modelling of a part of the glycolysis and the pentose phosphate pathway in erythrocytes.

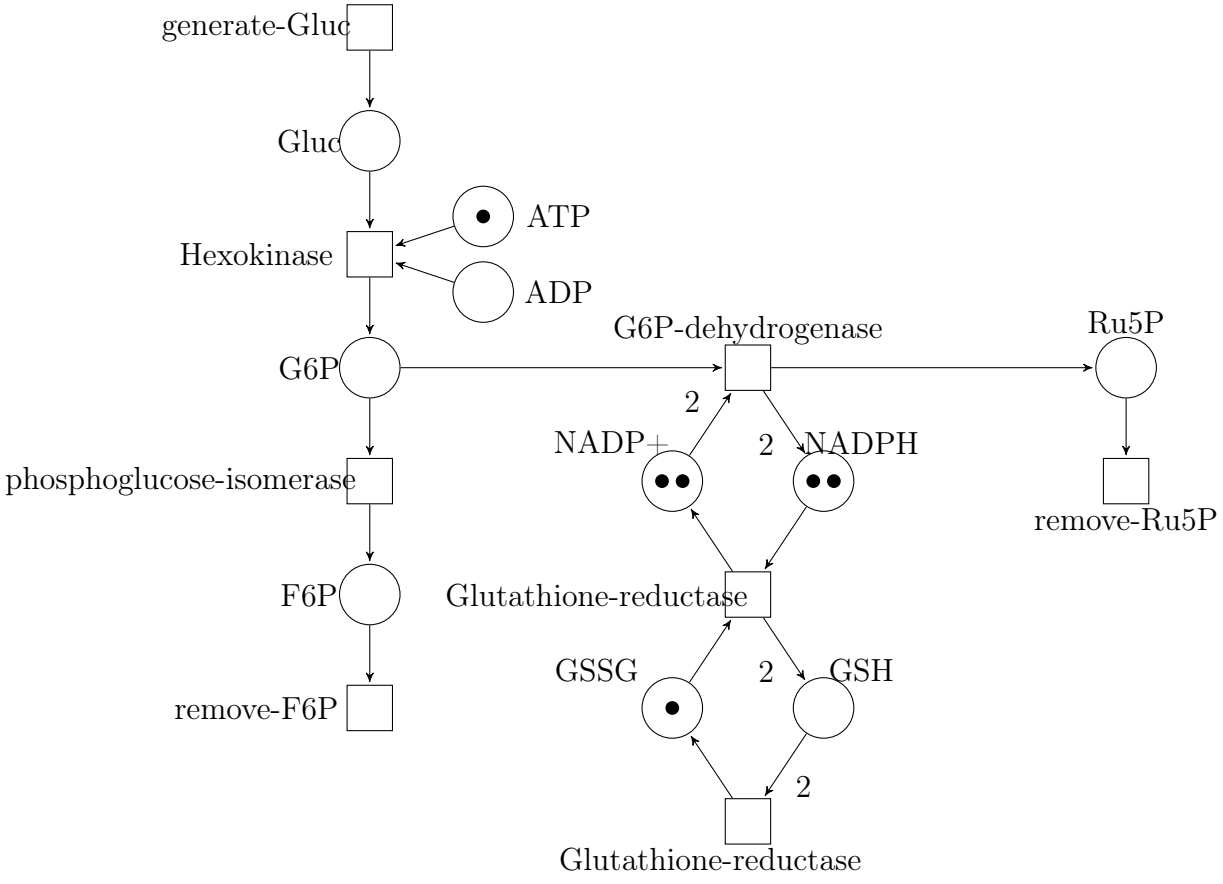



Figure 2.3: Petri net modelling a part of the glycolysis and the pentose phosphate pathway in erythrocytes

## 2.2 Biochemical Networks modelling

Biochemical networks can be interpreted as complex graphs modelled to understand the relationship between genes, proteins and metabolites in the cell. Modelling frameworks can be categorised based on what sort of information they include: continuous and/or discrete variables, static or dynamic model (taking time into account) and spatial features (consider the physical location molecules in the cell). The choice of framework depends on the purpose of our model (e.g. explanation of observed behaviour, prediction).

Reaction rule-based languages such as Biocham [9] provide several formal semantics for diagrams of biochemical interactions at different levels of abstraction:

- the stochastic semantics, in which the reaction rules are interpreted by a continuous time Markov chain.
- the discrete semantics, in which the rules are interpreted by a Petri net;
- the Boolean semantics, in which the rules are interpreted by a concurrent asynchronous transition system, allowing reasoning on the presence or absence of molecules;
- the differential semantics, in which the rules are interpreted by a system of ordinary differential equations.

It has been shown in [34], using the theory of abstract interpretation [25], how the different semantics of biochemical reaction rules, with the noticeable exception of the differential semantics, can be formally related by simple Galois connections, i.e. by formal abstraction relationships. These theoretical results allow us to state, for instance, that if a behaviour is not possible in the boolean semantics it is not possible in the stochastic semantics for any kinetic expressions and kinetic parameter values.

### 2.2.1 Boolean and Discrete modelling

Boolean networks have been widely used in modelling gene regulation networks. The state of genes is described by binary (ON/OFF) variables, and the dynamic behaviour of each variable, that is, whether it will be ON or OFF at next moment, is governed by a Boolean function. It is the case of systems like Kohn's map modelling the mammalian cell cycle regulation [61], with about 500 nodes and the mammalian Pathway Logic of [33].

### 2.2.2 Continuous and stochastic Modelling

Quantitative models are essentially based on systems of Ordinary Differential Equations (ODEs) and aim to representing the system in a detailed way producing quan-

titative results. They require kinetic data such as equilibrium constants and reaction rates, which are often lacking. Let us consider a biochemical system formed by $m$ chemical reactions and a set of chemical species $\{A_1, A_2, .., A_n\}$ that take part in the reactions (set of reactants union the set of products). The $j^{th}$ chemical reaction of the biochemical network is denoted as follows:

$$R_j\colon \sum_{i=1}^{n} S_{ij} A_i \Rightarrow \sum_{i=1}^{n} S'_{ij} A_i$$

where $S_{ij}$ and $S'_{ij}$ are positive integers called the stoichiometry coefficients. The right arrow means that the transformation of reactants into products only happens in the direction of the arrow. The compounds before this arrow are the reactants, and the ones after are the products. If the reaction is reversible, then we need to list its inverse in the chemical reaction network as a separate reaction. Each chemical reaction takes place continuously in time with its own rate which is a function of the concentration of the species taking part in it. More precisely, we define the vector $A = [[A_1], [A_2], .., [A_n]]$ of species concentrations and, as a function of it, we consider the vector of reaction rates $e(A) = [e_1, e_2, .., e_m]$. As an example, we cite the Mass-Action (MA) kinetics:

$$e_j = k_j \times \prod_i [A_i]^{S_{ij}}$$

Where $k_j$ is the equilibrium constant of the reaction $j$.Then, the concentration of the compound $A_i$ varies as follow:

$$\frac{d[A_i]}{dt} = \sum_{j=1}^{m} (S'_{ij} - S_{ij}) e_j$$

### 2.2.3 Petri nets modelling of biochemical networks

Petri nets have already been applied to biological case studies like the regulation of the lac operon [89], Duchenne muscular dystrophy [46], the response of S. cerevisiae to copulatory hormones [31] and the yeast cycle [77]. Two examples where they have been used for metabolic networks case studies, we find the sucrose breakdown pathway in the potato tuber [60] and the iron homoeostasis process in human body [88].

#### Standard Petri nets

The correspondence place/substance and transition/reaction or enzyme is apparent. In graphical representations of Petri nets, circles are used for places, while rectangles stand for transitions. The stoichiometric coefficients indicating how many molecules of a substance have to react to produce how many molecules of product correspond

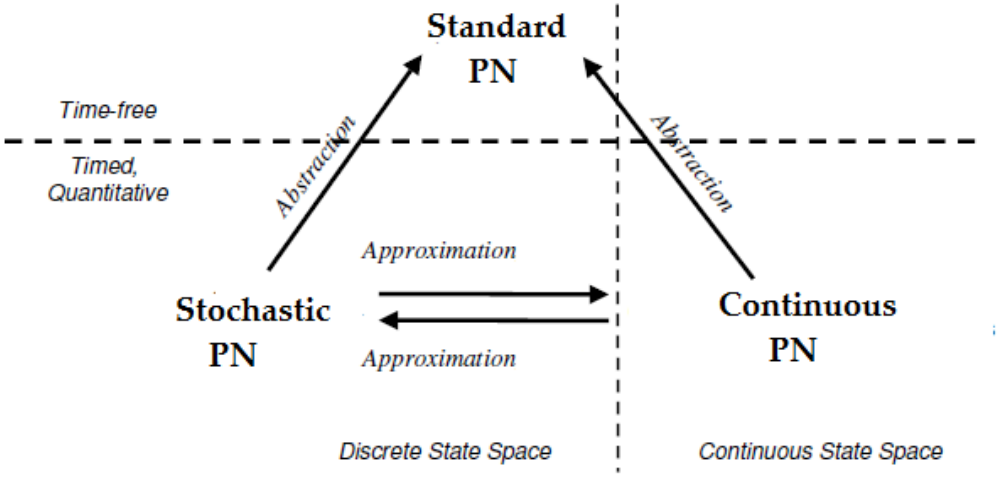
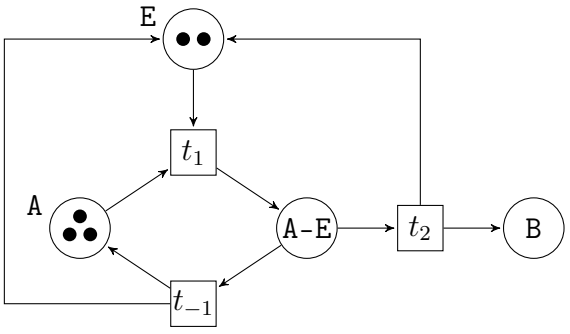



Figure 2.4: Conceptual Framework. The Petri net formalism allows to switch between different network classes to describe standard (qualitative) Petri nets, stochastic (SPN) and continuous (CPN) information in a cohesive Petri net model [47].

in Petri net formalism to arc weights. Thus, the stoichiometry matrix containing these coefficients corresponds to the incidence matrix of a Petri net. In our case, the number of tokens in a place stands for the number of molecules of that metabolite existing at a given moment. Thus, tokens may correspond to any predefined unit measuring the amount of substance, such as mole, millimole etc.

**Example 16.** *For instance the enzymatic reaction written (in BIOCHAM-like syntax),* `A + E <=> A-E => B + E` *corresponds to the following Petri net :*




### Stochastic Petri nets

Stochastic Petri nets (SPNs) emerged as a modelling formalism for performance analysis in the early 1980s. Analogous to the extension from classical process algebras to stochastic process algebras, stochastic Petri nets associate an *exponentially distributed delay* with the firing of each transition. The delay occurs between when

the transition becomes enabled (i.e. sufficient tokens arrive on its predecessors) and when it fires: the instantaneous firing will only occur if the transition has remained enabled throughout the delay period. As with standard Petri nets, stochastic Petri nets maintain a discrete number of tokens on places. But contrary to the time-free case, a firing rate (waiting time) is associated with each transition $t$, which are random variables $X_t \in [0, \infty)$, defined by probability distributions. Each transition gets its own local timer. When a particular transition becomes enabled , then the local timer is set to an initial value, which is computed at this time point by means of the corresponding probability distribution. In general, this value will be different for each simulation run. The local timer is then decremented at a constant speed, and the transition will fire when the timer reaches zero. If there is more than one enabled transition, a race for the next firing will take place. They have been applied to a gene regulatory network modelling since 1998 with Goss and Peccoud [43]. More recent applications of SPNs to biochemical networks are discussed in [97, 71].

A formal definition is the following:

**Definition 49** (Stochastic Petri nets (SPN))**.** *[41] An SPN is tuple* $SPN_{Bio} = (P, T, f, v, m_0)$*, where:*

- $P$*, *$T$* are finite, non empty, disjoint sets.* $P$ *is the set of places.* $T$ *is the set of transitions.*
- $f : ((P \times T) \cup (T \times P)) \to \mathbb{N}_0$ *defines the set of directed arcs, weighted by non-negative integer values.*
- $v : T \to H$ *is a function, which assigns a stochastic hazard function* $h_t$ *to each transition* $t$*, whereby* $H := \bigcup_{t \in T} \{h_t \mid h_t : \mathbb{N}_0^{|^\bullet t|} \to \mathbb{R}^+\}$ *is the set of all stochastic hazard functions, and* $v(t) = h_t$ *for all transitions* $t \in T$*.*
- $m_0 : P \to \mathbb{N}_0$ *gives the initial marking.*

Colored Petri nets were first proposed by Jensen [55]. In colored Petri nets, tokens are distinguished by the "color", rather than having only the "black" one. Besides, arc expressions, an extended version of arc weights, specify which tokens can flow over the arcs, and guards that are in fact Boolean expressions define additional constraints on the enabling of the transitions [56]. Formally, colored Petri net are as the following:

**Definition 50** (Colored Petri nets)**.** *[69] A colored Petri net is a tuple* $(P, T, F, \sum, C, g, f, m_0)$*, where:*

- $P$ *is a finite, non-empty set of places.*
- $T$ *is a finite, non-empty set of transitions.*
- $F$ *is a finite set of directed arcs, such that* $F \subseteq (P \times T) \cup (T \times P)$*.*
- $\sum$ *is a finite, non-empty set of types, also called color sets.*

- $C : P \to \sum$ *is a color function that assigns to each place* $p \in P$ *a color set* $C(p) \in \sum$.
- $g : T \to EXP$ *is a guard function that assigns to each transition* $t \in T$ *a guard expression that has the Boolean type.*

**Continuous Petri nets**

An other extension is the continuous case, in a continuous Petri net the marking of a place is no longer an integer, but a positive real number, called token value, which we are going to interpret as the concentration of the species modelled by the place. The instantaneous ring of a transition is carried out like a continuous flow.

**Definition 51** (Continuous Petri nets (CPN))**.** *[41] A CPN is tuple* $CPN_{Bio} = (P, T, f, v, m_0)$*, where:*

- $P$*,* $T$ *are finite, non empty, disjoint sets.* $P$ *is the set of continuous places.* $T$ *is the set of continuous transitions.*
- $f : ((P \times T) \cup (T \times P)) \to \mathbb{R}_0^+$ *defines the set of directed arcs, weighted by non-negative real values*
- $v : T \to H$ *is a function, which assigns a firing rate function* $h_t$ *to each transition* $t$*, whereby* $H := \bigcup_{t \in T} \{h_t \mid h_t : \mathbb{R}^{|^\bullet t|} \to \mathbb{R}^+\}$ *is the set of all firing rate functions, and* $v(t) = h_t$ *for all transitions* $t \in T$.
- $m_0 : P \to \mathbb{R}_0^+$ *gives the initial marking.*

## 2.3 Benchmark

Our main benchmark corresponds to the curated biological models of the biomodels.net[66] database. However, to compare to work from the Petri net community, we were led to consider also non biological models. We consider the 80 Petri nets from the Petriweb[45] database describing general industrial processes.

### 2.3.1 Biomodels.net

For computational modelling to become more widely used in biological research, researchers must be able to exchange and share their results. The development and broad acceptance of common model representation formats such as SBML is a crucial step in that direction, allowing researchers to exchange and build upon each other's work with greater ease and accuracy.

The biomodels.net[66] project is another step to:

1. define agreed-upon standards for model curation;

2. define agreed-upon vocabularies for annotating models with connections to biological data resources;

3. provide a free, centralized, publicly-accessible database of annotated, computational models in SBML and other structured formats.

Biomodels.net Database is a repository of peer-reviewed, published, computational models. These mathematical models are primarily from the field of systems biology, but more generally are those of biological interest. This resource allows biologists to store, search and retrieve published mathematical models. In addition, models in the database can be used to generate sub-models, can be simulated on line, and can be converted between different representational formats. This resource also features programmatic access via Web Services.

All unmodified models in the database are available freely for use and distribution, to all users. In addition, we consider the following complex biochemical models:

- Schoeberl's model of the MAP kinase cascade activated by surface and internalized EGF receptors [92] contains 100 places and 242 transitions.

- Calzone et al. E2F/Rb [10] has 408 places and 534 transitions.

- Kohn's map of the mammalian cell cycle control [61, 15], a model of 509 species and 775 reactions.

### 2.3.2 Petriweb

The Petriweb[45] database includes a set of Petri nets modelling industrial processes. The repository can be browsed with a web browser, and individual nets can be retrieved and uploaded in PNML, an emerging standard format supported by many tools.

The repository contains 80 Petri nets with associated properties. Properties are defined by the repository administrator. A property can be defined to be computed automatically from the net's PNML source through an external software program.

Petriweb is still in the prototype phase. It supports a restricted form of PNML, including flat, uncoloured nets, plus limited support for hierarchy. The database only contains a handful of examples.

## 2.4 Petri net properties on the benchmark

### 2.4.1 P-invariants as mass conservation laws

A Petri net's incidence matrix corresponds to the stoichiometric matrix in a metabolic network. The incidence matrix comprises the change in token amount for each place when a single transition of the whole network is fired as it is shown is Example 2. A P-invariant is a non-zero and non-negative integer place vector such that $VA = 0$, where $A$ is the incidence matrix. In the biological interpretation, P-invariants correspond to substance conservation that holds whatever the kinetics of the different reactions.

**Example 17.** *In the net depicted in Figure 2.3,*

$$m(ATP) + m(ADP) = 1 \tag{2.1}$$
$$m(NADP+) + m(NADPH) = 2 \tag{2.2}$$
$$2m(GSSG) + m(GSH) = 2 \tag{2.3}$$

### 2.4.2 T-invariants as flux conservation

T-invariants are solutions of $AV = 0$. A T-invariant has two interpretations in the given biochemical network.

1. The entries of a T-invariant represent a multiset of transitions, which reproduce a given marking by their partially ordered firing. That means that they occur basically one after the other. The partial order sequence of the firing events of the T-invariant's transitions may contribute to a deeper understanding of the system behaviour.

2. The entries of a T-invariant may also be read as the relative firing rates of transitions, all of them occurring permanently and concurrently. This activity level corresponds to the steady state behaviour.

The non-empty solution space of $AV = 0$ is infinite and so only minimal non-negative integer solutions are considered. A T-invariant is minimal if its support is not included in the support of an other T-invariant, and the greatest common divisor of all entries of the invariant vector is equal to 1. The support of a T-invariant contains the set of transitions with a positive entry. T-invariants have been studied a lot for the flux analysis of metabolic systems. Minimal T-invariants correspond to elementary modes. We use T-invariants in a different way for steady state analysis. Indeed, if a T-invariant defines a steady flux, it also provides an equation such that if one finds a state where all kinetics are those of this flux, it is a steady state. Noticing that it is not necessary to know the precise kinetics to solve such an equation, but only to make some hypotheses like "this is a Mass Action Law (MAL) kinetics"or "that is a Michaelis Menten (MM) kinetics".

### 2.4.3 Siphons/Traps

One example of the relevance of traps and siphons in biology was given in [107] for the analysis of the potato plant that produces starch and accumulates it in the potato tubers during growth, while starch is consumed after the tubers are deposited after the harvest. The starch and several of its precursors then form traps in the reaction net during growth, while starch and possible intermediates of degradation form siphons after the harvest.

The underlying Petri net is shown in Figure 2.5, where $G_1$ stands for glucose-1-phosphate, $G_u$ is UDP-glucose, $S$ is the starch and $I$ for intermediary species [99]. In this model, either the branch producing starch ($t_3$ and $t_4$) or the branch consuming it ($t_5$ and $t_6$) is operative. $P_1$ and $P_2$ represent external metabolites.

It can be easily observed that the set $\{G_u, S\}$ is a trap when $t_3$ and $t_4$ are operative: once a token arrives in $S$, no transition can be fired and the token remains there independently of the evolution of the system. Dually, $\{S, I\}$ is a siphon when $t_5$ and $t_6$ are operative: once the last token is consumed from $S$ and $I$, no transition can generate a new token in these places, so they remain empty.

In most cells containing starch, starch and specific predecessors form traps, whereas starch and specific successors form siphons. This provides a very simple explanation for the fact that either the branch producing starch or the branch degrading it is operative. This is realized by complete inhibition of the appropriate enzymes by the gene regulatory network.

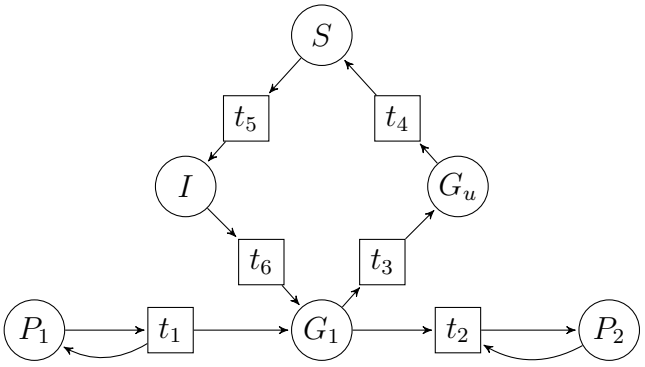



Figure 2.5: Petri-net graph modelling the growth metabolism of the potato plant [107].

Another interesting example, also from [107], deals with the analysis of the role of the triosephosphate isomerase (TPI) in Trypanosoma brucei metabolism by detecting solely siphons and traps. At the beginning, Helfert et al. [49] supposed that glycolysis could proceed without TPI. But unexpected results where all system fluxes (Pyruvate, Glycerol) decrease were found so that the authors built a kinetic model for explaining that phenomenon. Then a purely structural explanation for the necessary presence of TPI in glycolysis and glycerol production was provided in [107] by simply considering the presence of siphons and traps in the model.

# Chapter 3

# Boolean Model for siphons/traps

## Contents



## Introduction

A constraint satisfaction problem (CSP) consists of a set of variables, for each variable, a finite set of possible values (its domain) and a set of constraints restricting the values that the variables can simultaneously take. A solution to a CSP is an assignment of a value from its domain to every variable, in such a way that every constraint is satisfied.

## 3.1 Constraint Programming (CP) and Systems Biology

Constraint modelling [4] offers efficient tools to model real problems. It has been applied to many fields [104] like planning, resource allocation, computer networks,

bioinformatics and even bin packing.

In a constraint program, the user specifies a number of constraints. Each constraint defines a relation between the variables that describe the state of the studied system. The constraint programming tool provides constraint solving algorithms which infer new constraints from given ones and which compute solutions, i.e., valuations of the variables satisfying all the constraints.

Constraint Logic Programming is a successful merge between logic programming (LP) and constraint solving. Logic programming is based on predicates, unification, backtracking and depth-first search. Constraints can be seen as predicates describing relations between variables and constraint solving can be seen as a general form of unification which makes them compatible with logic programming.

In some cases, one can have powerful heuristics for solving a CSP, the heuristic can become less effective when we change the model. There is a large set of biochemical problems that we can prove they are intractable (NP-complete or worse) even with simplifications.

There are many motivations for using CSP in solving problems from biology, mainly, the fact that models are rarely stable and may change quickly and modifying a CSP model is not difficult. Constraint programming methods have been already applied to discover efficiently the steady-state of large gene regulation networks [28]. Fanchon and al. use constraints to infer ranges of parameter values from observations [38] and for analysing discrete genetic regulatory networks [20]. Chabrier and Fages [13] describe a model-checking approach with linear arithmetic constraints to check properties of qualitative or quantitative systems expressed in Computation Tree Logic. Bockmayr and Courtois [6] use Hybrid Concurrent Constraint to model a variety of biological phenomena, such as reaching thresholds, kinetics, gene interaction or biological pathways. In [63], Larhlimi and Bockmayr introduce a new approach to metabolic pathway analysis, characterizing a metabolic network by its minimal metabolic behaviours and the reversible metabolic space. Their method uses an outer description of the steady-state flux cone, based on sets of irreversible reactions. This is different from existing approaches, such as elementary flux modes or extreme pathways, which use an inner description, based on sets of generating vectors. The resulting description of the flux cone is minimal, unique, and satisfies a simplicity condition similar to the one that holds for elementary flux modes.

## 3.2 Boolean Model

In the literature, many algorithms have been proposed to compute minimal siphons and traps of Petri-nets. Since a siphon in a Petri-net $N$ is a trap of the dual net $N'$, it is enough to focus on siphons, the traps are obtained by duality. Some algorithms are based on linear programming [78, 21], Horn clause satisfaction [58, 75] or algebraic approaches [64]. More recent state-of-the-art methods are presented in [22, 23].

Here we present two Boolean methods for enumerating minimal siphons. First,

siphons can be straightforwardly characterized with a Boolean model representing the belonging or not of each place to the siphon. For a Petri-net with $n$ places and $m$ transitions, a siphon $S$ is a set of places whose predecessors are also successors. $S$ can be represented with a vector $\vec{V}$ of $\{0,1\}^n$ such that for all $i \in \{1,2,..,n\}$, $V_i = 1$ if and only if $p_i \in S$. The siphon constraint can then be formulated as:

$$\forall i, V_i = 1 \Rightarrow {}^\bullet p_i \subseteq (\bigcup_{V_j=1} \{p_j\})^\bullet$$

This constraint is equivalent to:

$$\forall i, V_i = 1 \Rightarrow (\forall t \in T, t \in {}^\bullet p_i \Rightarrow t \in (\bigcup_{V_j=1} \{p_j\})^\bullet)$$

is equivalent to:

$$\forall i, V_i = 1 \Rightarrow (\forall t \in T, t \in {}^\bullet p_i \Rightarrow \exists p_j \in {}^\bullet t, V_j = 1)$$

which can be rewritten again in clausal form as:

$$\forall i, V_i = 1 \Rightarrow \bigwedge_{t \in {}^\bullet p_i} (\bigvee_{p_j \in {}^\bullet t} V_j = 1)$$

To exclude the case of the empty set, the following constraint is added:

$$\bigvee_i V_i = 1$$

.

These clauses are Horn-dual clauses. They are trivially satisfied by taking all variables true.

Second, the enumeration of all minimal siphons (*w.r.t.* set inclusion) can be ensured by a search strategy and the addition of new Boolean constraints during search. One strategy is to find siphons in set inclusion order, and to add a new constraint

$$\bigvee_{p_i \in S} V_i = 0$$

each time a siphon $S$ is found to disallow any superset of this siphon to be found in the continuation of the search. It is worth remarking that this clause is not the dual of a Horn clause. The whole clauses are thus now non-Horn.

In a previous approach based on Constraint Logic Programming [79], the enumeration by set inclusion order was ensured by labelling a cardinality variable in

increasing order. Labelling directly on the Boolean variables, with increasing value selection (first 0, then 1), reveals however much more efficient and in fact easier to enforce. The following proposition shows that this strategy correctly finds siphons in set inclusion order.

**Proposition 12.** *Given a binary tree such that, in each node instantiating a variable $X$, the left sub-edge posts the constraint $X = 0$ and the right sub-edge posts the constraint $X = 1$, then for all distinct leaves $A$ and $B$, leaf $A$ is on the left of leaf $B$ only if the set represented by $B$ is not included in the set represented by $A$ (that is to say, there exists a variable $X$ such that $X_B > X_A$, where $X_A$ and $X_B$ denote the values instantiated to $X$ in the paths leading to $A$ and $B$ respectively).*

*Proof.* $A$ and $B$ have a least common ancestor node instantiating a variable $X$. If leaf $A$ is on the left of leaf $B$, the sub-edge leading to $A$ is the left one, with the constraint $X = 0$ and the sub-edge leading to $B$ is the right one, with the constraint $X = 1$, therefore $X_B > X_A$. □

In a post-processing phase, the computed set of minimal siphons can be filtered for only keeping the minimal siphons that contain a given set of places, and hence solve the above mentioned NP-complete decision problem of existence of minimal siphon containing a given set of places (Rec-MSEP). It is worth remarking that posting the inclusion of the selected places first would not ensure that the siphons found are indeed minimal w.r.t. set inclusion.

## 3.3 Boolean Algorithms

This section describes two implementations of the above model and search strategy, one using an iterated SAT procedure and the other based on Constraint Logic Programming with Boolean constraints.

### 3.3.1 Iterated SAT Algorithm

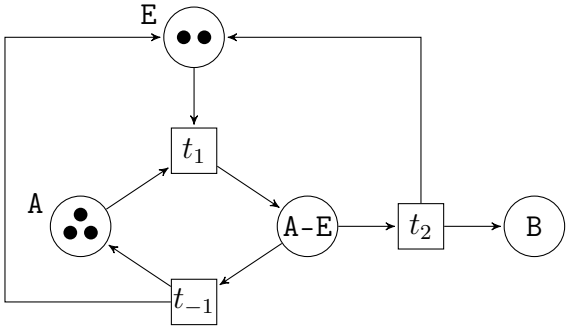
The Boolean model can be directly interpreted using a SAT solver to check the existence of a siphon or trap. We use sat4j[1], an efficient library of SAT solvers in Java for Boolean satisfaction and optimization. It includes an implementation of the MiniSAT algorithm in Java. Like many SAT solvers, MiniSAT requires that its input be in conjunctive normal form (CNF). Given a siphon S, for each place p belonging to it, the predecessors of p is a subset of the set of successors of S. For each transition in the set of predecessors of p, a clause C is added to the satisfiability problem. C is formed by a negated Boolean variable associated to p and Boolean variables in the set of predecessors of t. These terms are connected with or operator.

[1] http://www.sat4j.org/

To avoid the the trivial case of empty siphon, we add a quite simple clause formed by the n Boolean variables in their negated form. To find the whole set of minimal siphons, we iterate on the resolution of this problem adding a clause of minimality at each iteration: this clause avoids finding the already found siphon and all siphons containing it. This means we add a clause formed by Boolean variables present in the found siphon under their negated form. That means that at least one of these variables is equal to zero.

The example of the enzymatic reaction of example 16 is encoded as follows:

Let us consider again the figure of the example 16:




The Boolean variables $x_1$, $x_2$, $x_3$ and $x_4$ correspond respectively to $E$, $A$, $AE$ and $B$. In the first iteration, the problem amounts to solve the following encoding of Horn-dual clauses:
$\neg x_2 \vee x_3$
$\neg x_3 \vee x_1 \vee x_2$
$\neg x_1 \vee x_3$
$\neg x_1 \vee x_3$
$\neg x_4 \vee x_3$
The problem is satisfied with the values: $x_1 = x_4 = 0$ and $x_2 = x_3 = 1$, which means that $\{A, AE\}$ is a minimal siphon.

To ensure minimality, the (non Horn-dual) clause $x_2 = 0 \vee x_3 = 0$ is added and the program iterates an other time. The problem is satisfied with $x_2 = x_4 = 0$ and $x_1 = x_3 = 1$ meaning that $\{E, AE\}$ is also a minimal siphon. A new clause is added stating that either $E$ or $AE$ does not belong to the siphon and no more variable assignment can satisfy the problem.

Therefore, this model contains 2 minimal siphons: $\{A, AE\}$ and $\{E, AE\}$. The enzyme $E$ is a catalyst protein for the transformation of the substrate $E$ in a product $B$. Such a catalyst increases the rate of the reaction but is conserved in the reaction.

### 3.3.2 Backtrack Replay CLP(B) Algorithm

The search for siphons can also be implemented with a Constraint Logic Program with Boolean constraints (CLP(B)). We use GNU-Prolog[2] [29], for its efficient low-level implementation of Boolean constraint propagators.

The enumeration strategy is a variation of *branch-and-bound*, where the search is restarted to find a non-superset siphon each time a new siphon is found. We tried two variants of the branch-and-bound: with restart from scratch and by backtracking.

In the branch-and-bound with restart method, it is essential to choose a variable selection strategy which ensures diversity. Indeed, an enumeration method with a fixed variable order accumulates failures by always trying to enumerate the same sets first and these failures are only lately pruned by the non-superset constraints. As a consequence, the search tree developed gets more and more dense after each iteration since the previous forbidden sets are repeatedly tried again. This phenomenon does not exist in SAT solvers thanks to no-good recording. In CLP, this problem can be compensated for however, by using a random selection strategy for variables. This provides a good diversity and performs much better than any uniform heuristics.

However, branch-and-bound by backtracking gives better performance when care is taken for posting the non-superset constraint only once, since reposting it at each backtrack step proved to be inefficient. Our backtrack replay strategy is implemented as follows:

1. each time a siphon is found, the path leading to this solution is memorized,

2. then the search is fully backtracked in order to add to the model the new non-superset constraint,

3. and then the memorized path is rolled back to continue the search at the point it was stopped.

Figure 3.1, generated with CLPGUI[3] [36], depicts the search tree that is developed for enumerating the 64 minimal siphons of a biological model of 51 species and 72 reactions. Each sub-tree immediately connected to the root corresponds to the replay of the path with a minimality constraint added. It is remarkable that with the backtrack replay strategy, very few backtracking steps are necessary to search for all solutions.

[2] http://www.gprolog.org/
[3] http://contraintes.inria.fr/~fages/CLPGUI

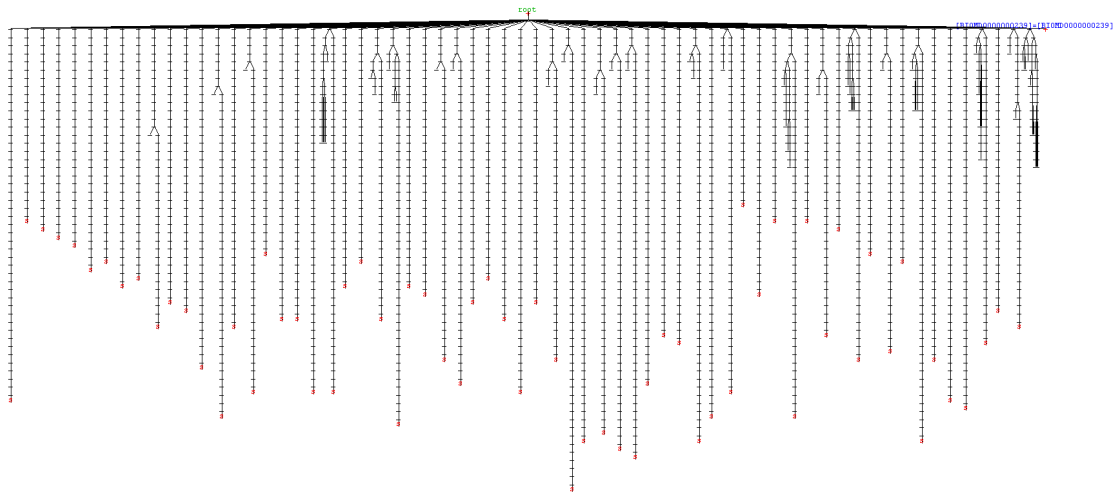


Figure 3.1: Search tree developed with the backtrack replay strategy for enumerating the 64 minimal siphons of model 239 of biomodels.net (described in Section 2.3). Each red end corresponds to a minimal siphon found. Very few backtracks are necessary thanks to the constraint propagation and the strategy

## 3.4 Evaluation

### 3.4.1 Results and Comparison

Our evaluation is carried out on the benchmark presented in 2.3. In this section, we compare the two Boolean methods described in the previous section with the state-of-the-art dedicated algorithm of [23]. Pseudo-code of the state-of -the-art algorithm is presented in appendix A. This algorithm uses a recursive problem partitioning procedure to reduce the original search problem to multiple simpler search sub-problems. Each sub-problem has specific additional place constraints with respect to the original problem. This algorithm can be applied to enumerate minimal siphons, place-minimal siphons, or even siphons that are minimal with respect to a given subset of places.

| Database | # model | # siphons min-max (avg.) | siphons size min-max (avg.) | total time (ms) | | |
|---|---|---|---|---|---|---|
| | | | | dedicated algorithm | SAT | GNU Prolog |
| Biomodels.net | 403 | 0-64 (4.21) | 1-413 (3.10) | 19734 | 611 | 195 |
| Petriweb | 80 | 0-11 (2.85) | 0-7 (2.03) | 2325 | 156 | 6 |

Table 3.1: Computation time in milliseconds on the biomodels and Petriweb benchmarks.

Table 3.1 presents the CPU times in milliseconds for enumerating all minimal siphons of the Petri nets in Petriweb and biomodels.net. All times are in milliseconds

and have been obtained on a PC with an intel Core processor 2.20 GHz and 8 GB of memory. For each benchmark, we provide the total number of models, the minimal, maximal and average numbers of siphons and the total computation time in milliseconds for enumerating all of them.

Surprisingly, but happily, on all these practical instances, except one instance detailed below, the SAT and CLP(B) programs solve the minimal siphon enumeration problem, in less than one millisecond in average, with a better performance for the CLP(B) program over the SAT solver, and by two orders of magnitude over the dedicated algorithm.

| model | # siphons | # places | # transitions | dedicated algorithm | sat | GNU Prolog |
|---|---|---|---|---|---|---|
| Kohn's map of cell cycle | 81 | 509 | 775 | 28 | 1 | 221 |
| BIOMD000000175 | 3042 | 118 | 194 | $\infty$ | 137000 | $\infty$ |
| BIOMD000000205 | 32 | 194 | 313 | 21 | 1 | 34 |
| BIOMD000000239 | 64 | 51 | 72 | 2980 | 1 | 22 |

Table 3.2: Computation time in milliseconds on the hardest instances of biochemical networks.

However, one particular model, number 175 in biomodels.net, was excluded from this table because its computational time is very high. Table 3.2 presents the performance figures obtained on this model and on the three other hardest instances for which we also provide the number of places and transitions. On these hard instances, the SAT solver is faster than the CLP(B) program by one to two orders of magnitude, and is the only algorithm to solve the problem for model 175, in 137 seconds.

That model 175 represents a quantitative model that relates EGF and HRG stimulation of the ErbB receptors to ERK and AKt activation in MCF-7 breast cancer cells [5]. This is the first model to take into account all four ErbB receptors, simultaneous stimulation with two ligands, and both the ERK and AKt pathways. Previous models of ErbB (e.g. the model developed in [92]) were limited to a single ErbB because of combinatorial complexity. It is well known that the ErbB signaling network is highly connected and indeed the underlying Petri-net contains the highest number of arcs of the biomodels.net repository.

### 3.4.2 Hard instances

MiniSAT and CLP(B) outperform the specialized algorithm by at least one order of magnitude and the computation time is extremely short on our practical examples. Even if the model is quite large, e.g. for Kohn's map of the cell cycle control with 509 species and 775 reactions, the computation time for enumerating its 81 minimal siphons is astonishingly short: one millesecond only. However, this enumeration of all minimal siphons solves the Rec-MSEP problem which has been proved NP-

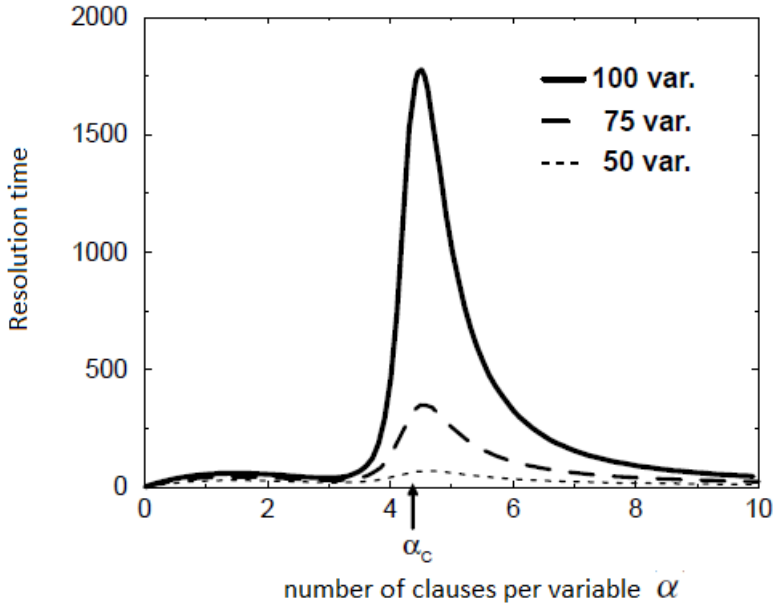



Figure 3.2: Computation time random 3-SAT

complete by reduction of 3-SAT in [106], and the question is: why are the CSP-based algorithms for enumerating siphons so efficient on the existing benchmarks from systems biology?

We can provide some hint of explanation by considering the well-known phase transition phenomenon in 3-SAT. The probability that a random 3-SAT problem is satisfiable has been shown to undergo a sharp phase transition as the ratio $\alpha$ of the number of clauses over the number of variables crosses the critical value of about 4.26 [76, 26] as it is shown in Figure 3.2 , going from satisfiability to unsatisfiability with probability one when the number of variables grows to infinity as it is shown in Figure 3.3.

Our first hint of explanation is by looking if the density of 3-SAT instances derived from general SAT instances of Biomodels.net are greater than the critical value of 4.26. If it is the case, we can estimate that if the density is above the critical value, instances are easy because there is a small number of solutions. On the other hand, if the density is below the threshold value of 4.26, that is for small values of $\alpha$, computation time is long because clauses are satisfiable with an exponential number of valuations.

Density of a SAT instance is:

$$\text{density} = \alpha = \frac{\#clauses}{\#variables}$$

Considering our problem of enumerating minimal siphons of a general Petri net $PN = (P, T, W)$ on our | P | Boolean variable, initially we have $\sum_{t \in T} \mid t^{\bullet} \mid$ clauses

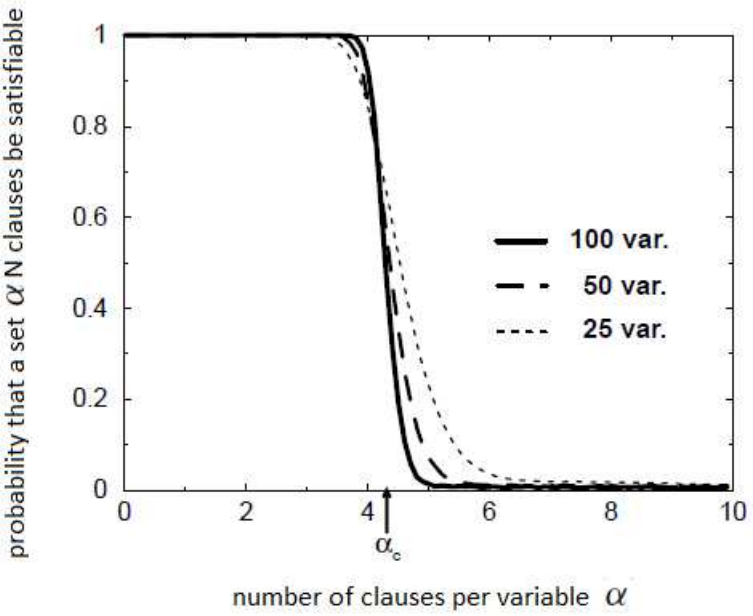



Figure 3.3: Probability of random 3-SAT satisfiablity

plus one clause of non-empty siphon:

$$\text{density} = \frac{\sum_{t \in T} \mid t^{\bullet} \mid +1}{\mid \mathrm{P} \mid}$$

To transform a general SAT instance to a 3-SAT instance, we add a number of clauses and a number of variables:

$$\text{density}_{\text{3-SAT}} = \frac{\sum_{t \in T} \mid t^{\bullet} \mid +1+\mu}{\mid \mathrm{P} \mid +\mu}$$

where

$$\mu = \sum_{t \in T} \max(0, \mid {}^{\bullet}t \mid -2)$$

The density distribution of Biomodels.net are illustrated in the histogram of figure 3.4.

This histogram shows that, contrary to our expectation, density is below the critical value for the majority of models. It is growing with enumeration (by adding minimality clauses), with possibility to reach the critical region of $\alpha$. The 3-SAT density of our hardest model number 175 equals 2.39. Hence the density is not a

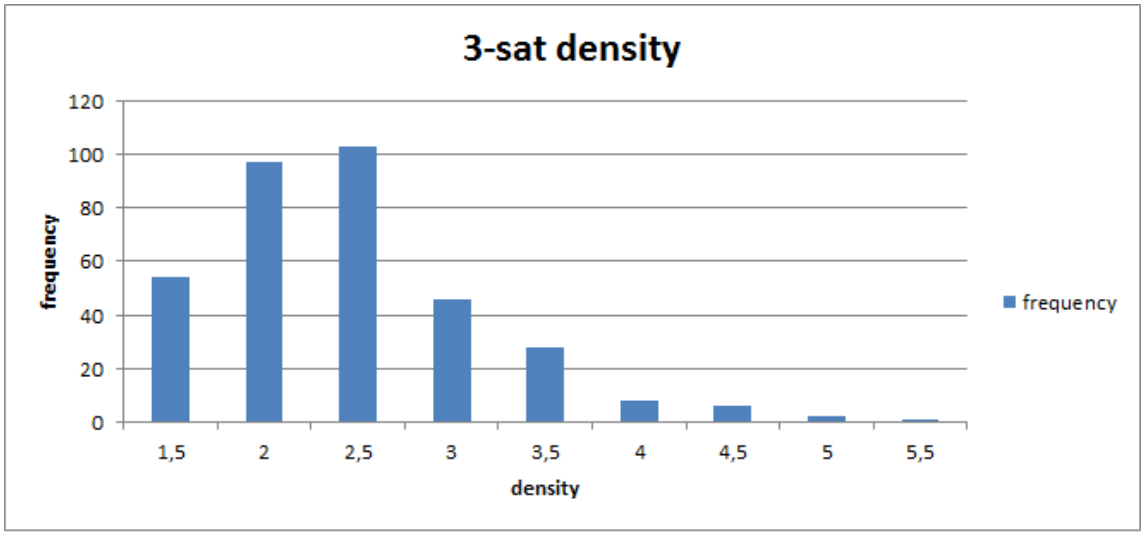



Figure 3.4: Distribution of density of 3-SAT models derived from Biomodels.net. Computed tree-widths are less or equal than 10.

sufficient measure to explain why we are so efficient in enumerating all solutions of an NP-complete problem.

Another insight of explanation recalls the linear time complexity result proved in section 1.3 for Petri nets with bounded tree-width: Rec-MSEP can be solved in linear time for Petri with bounded tree-width and biochemical models, despite of their large size, their corresponding Petri nets seem to have a small tree-width. We use the QuikBB algorithm to compute tree-width, this algorithm is available at http://graphmod.ics.uci.edu/group/quickbb. When given enough time, it yields the exact tree-width of the graph, when stopped before termination, it yields an upper bound on the tree-width. We applied QuickBB on the 432 models from the repository Biomodels.net version February 2013. Among these 432 models, 59 models do not hold any transitions. For 31 models, the exact tree-width could not be computed in a time-out of one hour, but we are sure that the upper bound is 23. For the remaining 342 models, the exact tree-width is computed and it is at most 10 as shown in Figure 3.5. We remark that tree-width remains less than 10 even for Petri nets of large size. For our 3 hard instances, tree-width are represented in Table 3.3, we could determine an upper bound equal to 15. Biomodels.net represent an easy benchmark for the enumeration of minimal siphons containing a given set of places. Our experimental results lead us to expect that our approach can be applied to models of tree-width less than 10. Hence, when the tree-width is less than 10, we expect no performance issues.

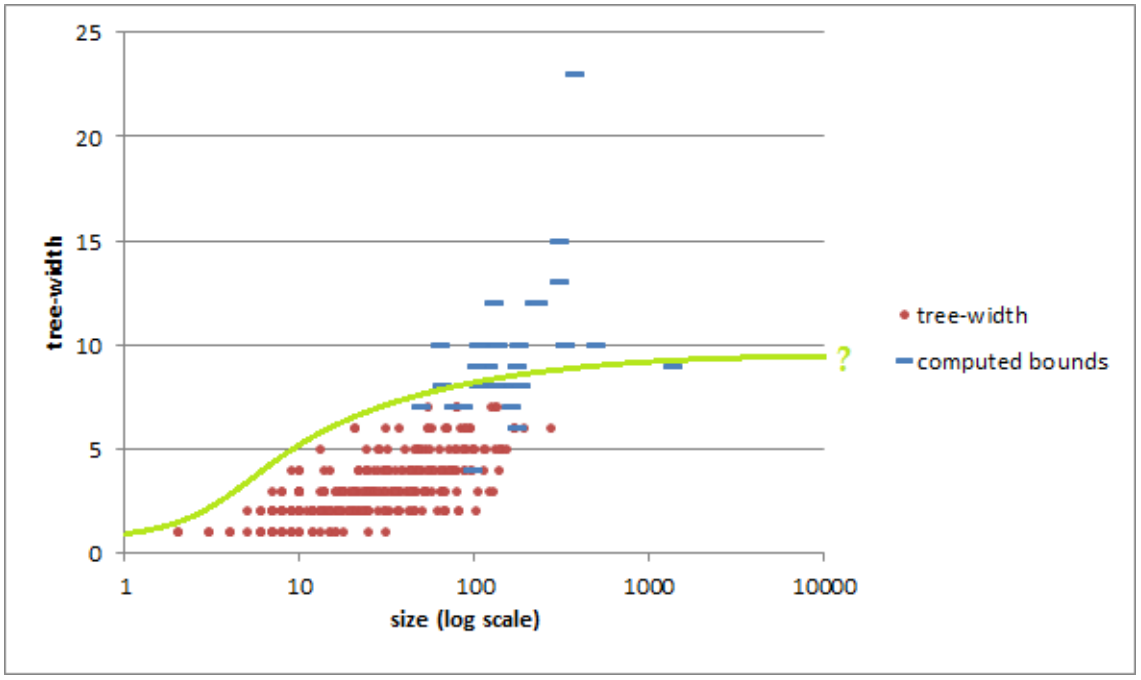



Figure 3.5: Variation of tree-width as a function of size (places and transitions) on Petri nets of Biomodels.net

| model | # places | # transitions | tree-width |
|---|---|---|---|
| BIOMD000000175 | 118 | 194 | $\leq 15$ |
| BIOMD000000205 | 194 | 313 | $\leq 10$ |
| BIOMD000000239 | 51 | 72 | $\leq 10$ |

Table 3.3: Tree-width of the hardest instances of Biomodels.net database.

## 3.5 CLP model for the Siphon-Trap Property (STP)

In [81], authors propose a reduction of STP to a SAT problem. STP holds when there is no siphon containing an unmarked trap. Authors aim at a formula that is satisfiable when there is a siphon which does not contain a marked trap. The starting point is the following formula whose satisfying assignments corresponds exactly to non empty siphons of a net:

$$\bigvee_{s \in S} s \wedge \bigwedge_{t \in T} \bigwedge_{s \in t^\bullet} (s \Rightarrow \bigvee_{s' \in {}^\bullet t} s').$$

If a siphon contains a marked trap, then its maximal included trap is necessary marked. Given a siphon $D$, its maximal included trap can be computed by a repeated removal of places $s$ from $D$: $s$ is removed when some of its successors are not predecessors of the so far remaining set. Let $n$ the number of places of $N$. Authors represent the repetition of the procedure by introducing $(n+1)$ variables $s^{(0)}, ..., s^{(n)}$ for each place $s$. The variables $s^{(0)}$ represent a non empty siphon. The variables

$s^{(i)}$ represent intermediate stages $D_i$ of the procedure for generating the maximal included trap. $D_{i+1}$ is obtained from $D_i$ by removing all places whose some successors are not predecessors of $D_i$. Since there are only $n$ places, the procedure converges after at most $n$ iterations: $D_n$ is either empty or the maximal trap included on $D$. The relation between $D_i$ and $D_{i+1}$ can be expressed as follows:

$$s^{(i+1)} \Leftrightarrow (s^{(i)} \wedge \bigwedge_{t \in T} \bigwedge_{s \in {}^\bullet t} \bigvee_{s' \in t^\bullet} s'^{(i)}).$$

And the formula is satisfied when the maximal trap is empty or unmarked. Hence the following formula is added:

$$\bigwedge_{s \in S : m_0 > 0} \neg s^{(n+1)}$$

From the above considerations the following theorem holds.

**Theorem 19.** *[81] In a given net $N$ with $n$ places, there exists a siphon which does not contain a marked trap if and only if the following formula is satisfiable:*

$$\phi ::= \bigvee_{s \in S} s \wedge \bigwedge_{t \in T} \bigwedge_{s \in t^\bullet} (s \Rightarrow \bigvee_{s' \in {}^\bullet t} s')$$
$$\wedge \bigwedge_{i=0}^{n} \bigwedge_{s \in S} (s^{(i+1)} \Leftrightarrow (s^{(i)} \wedge \bigwedge_{t \in T} \bigwedge_{s \in {}^\bullet t} \bigvee_{s' \in t^\bullet} s'^{(i)}))$$
$$\wedge \bigwedge_{s \in S : m_0 > 0} \neg s^{(n+1)}$$

If there is no satisfying assignment for $\phi$ then, all siphons contain a marked trap and the STP holds. In [81], experimental results done on $k$ dining philosophers show that MiniSat performs much better than the Petri net dedicated tool INA[87] that exploses since 50 philosophers [81]. An example of 2 philosophers is depicted in figure 3.6. The two philosophers alternatively think and eat. $p_0$ , $p_1$ are chopsticks. $p_2$, $p_3$ are philosophers thinking and $p_4$, $p_5$ are philosophers eating.

We encode the problem of Siphon-Trap property with a SAT and a CLP(FD) solver, we use respectively MiniSAT and GNU-prolog solver. The time needed by both solvers to decide about the existence of a minimal siphon containing an unmarked trap is shown in table 3.5. This experimental evaluation shows the high performance of Minisat solver: it solves each of the instances in less than one second. GNU-prolog solver outperforms MiniSAT on small instances (about 10 and 20 philosophers) but does not perform well on big instances. The high performance of the SAT solver is due to solving techniques like unit propagation, clause learning, conflict-directed backtracking, automated labelling heuristics and randomized restart. On the other hand CLP(FD) becomes less effective when the search space become big, it relies on propagation and search to solve a problem and the system does not learn from failures during search.

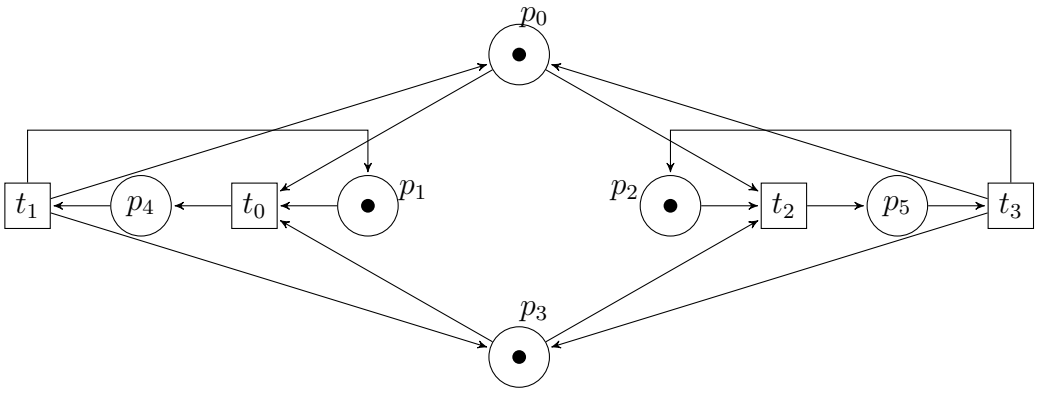



Figure 3.6: Petri net modelling the problem of 2 dining philosophers

| model | nb places | nb transitions | nb acrs | CPU time (sec) | |
|---|---|---|---|---|---|
| | | | | SAT | GNU Prolog |
| phils10 | 30 | 20 | 80 | 0.03 | 0.01 |
| phils20 | 60 | 40 | 160 | 0.07 | 0.03 |
| phils50 | 150 | 100 | 400 | 0.2 | 0.62 |
| phils100 | 300 | 200 | 800 | 0.29 | 4 |
| phils150 | 450 | 300 | 1200 | 0.46 | 14 |
| phils200 | 600 | 400 | 1600 | 0.65 | 31 |

Table 3.4: Siphon-Trap Property evaluation on $k$ dining philosophers Petri nets

# Chapter 4

# Constraint Programming Approach to P/T invariants

## Contents



## Introduction

In this chapter, we present a simple method due to Sylvain Soliman to extract minimal semi-positive invariants of a Petri net modelling a biological reaction system, as a constraint satisfaction problem on finite domains using constraint programming with symmetry detection and breaking. We evaluate the computation of minimal T-invariants and minimal P-invariants on our benchmark, already presented in the section 2.3. An implementation based on GNU-Prolog's FD solver of the method is incorporated in Biocham and in the Nicotine tool [94], a Constraint-Programming-based T and P-invariant Extractor. This tool[1] allows us import and export of several

[1] `http://contraintes.inria.fr/~soliman/nicotine`

formats (APNN, PNML, SBML, BIOCHAM, etc.).

In the second part of this chapter, we present a way to generalize well-known results about the steady-state analysis of some symbolic Ordinary Differential Equation systems by taking into account the structure of the reaction network. The structural study of the underlying Petri net will provide classes where the computation of some steady states of the system is possible, even though the original symbolic model did not form an S-system and was not solvable by state-of-the-art symbolic computation software. This new method is then illustrated on some models of the Biomodels.net repository. The method is implemented in the Nicotine tool.

## 4.1 P/T-invariants Computation

We recall the definitions of P and T-invariants as follows:

- A vector $V = [v_1, v_2, ..., v_n]$ with non-negative integer components is a P-invariant if $VA = 0$, where $A$ is the incidence matrix of the Petri net with $n$ places and $m$ transitions.
- A vector $Y = [y_1, ...y_m]$ with non-negative integer components is a T-invariant if $AY^t = 0$.

### 4.1.1 The Fourier-Motzkin Algorithm for P/T-invariants

The Fourier-Motzkin method is well known for computing a set of invariants including all minimal invariants [70, 19]. The Fourier-Motzkin algorithm to compute P-invariants is the following:

**Algorithm 1** Fourier Motzkin Elimination

1: Initialise B = [A : In]
2: **for** j = 1,..., m **do**
3: Append to B all rows resulting from positive linear combinations of pairs of rows in B that eliminate column j.
4: Remove from B all rows with non-zero $j^{th}$ element.
5: **end for**
6: B = [0 : D], where the rows of D are P-invariants.

This method has critical deficiencies such that, even if invariants exist, none of them may be computed because of a huge number of candidate vectors forcing the program to terminate before any invariant is generated. Moreover, even when a number of invariants are computed, many non minimal invariants may be included as it is shown in example 18 of the Petri net depicted in figure 4.1.

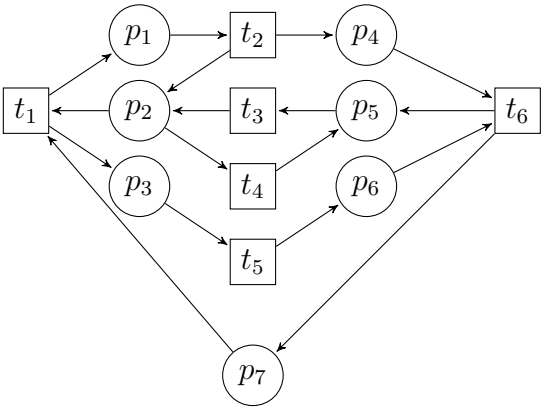



Figure 4.1: A Petri net example for Fourier-Motzkin algorithm

**Example 18.** *The incidence matrix of the Petri net depicted in Figure 4.1 is as follows:*

$$I = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

*Applying the Fourier-Motzkin elimination to compute p-invariants leads to the following transformation on the matrix $B$.*

$$B = \left[\begin{array}{cccccc|ccccccc} 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$B = \left[\begin{array}{cccccc|ccccccc} 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ & & & & & & & & & & & & \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & -1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$$

$$B = \left[\begin{array}{cccccc|ccccccc} 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ & & & & & & & & & & & & \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 2 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{array}\right]$$

$$B = \left[\begin{array}{cccccc|ccccccc} 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 2 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 2 & 1 \end{array}\right]$$

$$B = \left[\begin{array}{cccccc|ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 2 & 1 \end{array}\right]$$

*P-invariants are:*

- $x_1 = [1, 0, 0, 1, 0, 0, 1]$
- $x_2 = [0, 0, 1, 0, 0, 1, 1]$
- $x_3 = [1, 1, 0, 0, 1, 1, 0]$

- $x4 = [1, 1, 1, 0, 1, 2, 1]$

*It is obvious that $x_4$ is equal to the linear combination of $x_2$ and $x_3$ and is not a minimal P-invariant.*

Its worst case complexity is exponential due to a possible exponential growth in the number of matrix rows generated in the solution process. An implementation of the Fourier-Motzkin elimination procedure due to Peter Kemper is included in the Bio-PEPA tool suite in the Bio-PEPA Eclipse Plug-in.

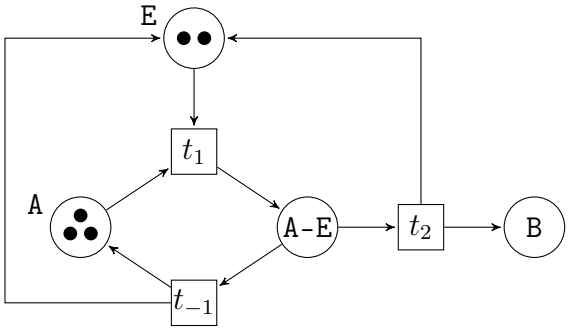
Many other algorithms based on Fourier-Motzkin elimination have been introduced aiming to decrease the number of candidate vectors by, for example, restricting computation of invariants to place sets that are siphons and traps at the same time [105].

Another way to extract the minimal semi-positive invariants of a model is to use one of the software tools that provide this computation for biological systems, generally as "conservation law"computation such as METATOOL [103] and COPASI [52] tools.

### 4.1.2 Finding P/T-invariants as a Constraint Solving Problem

This method is introduced in [93, 95], we illustrate the method for computing the invariants with the case of P-invariants. For a Petri net with $p$ places and $t$ transitions ($L_i \rightarrow R_i$), a P-invariant is a vector $V \in \mathbb{N}^p$ s.t. $V \cdot I = 0$, i.e. $\forall 1 \leq i \leq t \; V \cdot L_i = V \cdot R_i$. Since those vectors all live in $\mathbb{N}^p$, it is quite natural to see this as a Constraint Solving Problem (CSP) with $t$ (linear) equality constraints on $p$ Finite Domains variables.

**Example 19.** *Let us consider again the figure of Example 16:*




$$\begin{aligned} A + E &\Rightarrow A{-}E \\ A{-}E &\Rightarrow A + E \\ A{-}E &\Rightarrow B + E \end{aligned}$$

*This results in the following equations:*

$$A + E = AE \quad (4.1)$$
$$AE = A + E \quad (4.2)$$
$$AE = B + E \quad (4.3)$$

*where obviously equation (2) is redundant.*

The task is actually to find non trivial p-invariants with minimal support. To avoid the trivial case, the following constraint is added:

$$V \cdot \mathbf{1} > 0.$$

**Example 20.** *In our running example we thus add* $A + E + AE + B > 0$.

To ensure minimality, the author proceeds with the same branch and bound technique used for siphons and traps: the labelling is invoked from small to big values and a branch and bound procedure is wrapped around it, maintaining a partial set $\mathcal{S}$ of P-invariant vectors and adding the constraint that a new vector $V$ is solution if:

$$\forall S \in \mathcal{S} \prod_{S_i \neq 0} V_i = 0$$

which means that its support is not bigger than that of any vector already stored.

Unfortunately, even with the last constraint, no search heuristic was found that makes removing subsumed P-invariants unnecessary. Thus, if a new vector is added to $\mathcal{B}$, previously found vectors with a bigger support must be removed. The algorithm can be summarized as follows:

**Algorithm 2** Minimal P-invariants computation

1: post the CSP for invariant V: $\forall 1 \leq i \leq t \; V \cdot L_i = V \cdot R_i$ and $V \cdot \mathbf{1} > 0$
2: **repeat**
3: find a solution, enumerating from low to high
4: add the solution to the basis
5: remove non-minimal invariants from the basis if there are any
6: post the new constraint $\forall B \in \mathcal{B} \; \prod_{B_i \neq 0} V_i = 0$
7: **until** no solution found

This algorithm was implemented directly into Biocham [9], which is programmed in GNU-Prolog.

**Example 21.** *In our running example we find two minimal semi-positive P-invariants:*

- $E = AE = 1$ *and* $A = B = 0$
- $A = B = AE = 1$ *and* $E = 0$

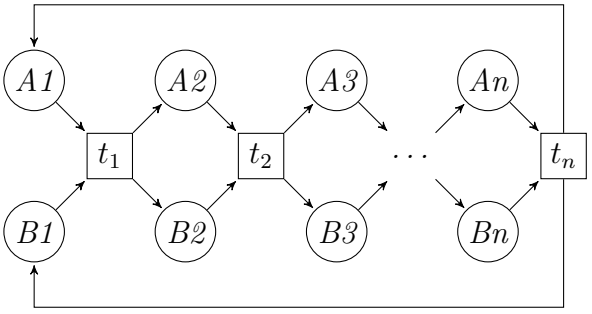
### 4.1.3 Symmetry detection and elimination

#### Complexity

The problem of computing all minimal semi-positive p-invariants is EXPSPACE-hard since there can be an exponential number of such structures. This is the case in the model given in Example 22 has $2^n$ minimal semi-positive P-invariants (each one with either $A_i$ or $B_i$ equal to 1 and the other equal to 0).

**Example 22.**

$$
\begin{aligned}
&A1 + B1 => A2 + B2\\
&A2 + B2 => A3 + B3\\
&\ldots\\
&An + Bn => A1 + B1
\end{aligned}
$$




#### Variable Symmetry detection and breaking

Equality classes optimization [65] for the standard Fourier-Motzkin algorithm corresponds to global and local variable symmetry detection and elimination for ordinary nets. In Example 22, there is a variable symmetry [40] between all the pairs $(A_i, B_i)$ of variables corresponding to places. This symmetry is easy to detect: variables corresponding to parallel places (places with the same set of predecessors and the same set of successors) are symmetric and can be eliminated through the usual ordering of variables, by adding the constraints $A_i \leq B_i$. In our prototype, we consider only global symmetry, that is, symmetry of the initial problem (i.e. the problem at the root of the search tree). In the method presented in [65], classes of equivalent variables are detected and eliminated also during the search which corresponds to local symmetry (symmetries that appear at each node of the search tree) and was not implemented in our prototype.

Moreover, in [65], equality class elimination is done through replacement of the symmetric places by a representative place. The full method reportedly improves by a factor two the computation speed. Even if in the context of the original article this is done only for ordinary Petri nets, we can see that it can be even more efficient to use this replacement technique in our case in order to become exponential as late as possible as it is shown in the following example:

**Example 23.**

$$
\begin{aligned}
&\ldots\\
&A + B => 4*C\\
&\ldots
\end{aligned}
$$

*Instead of simply adding $A \leq B$ to our constraints, which will lead to 3 solutions when $C = 1$ before symmetry expansion: $(A, B) \in \{(0, 4), (1, 3), (2, 2)\}$, replacing $A$ and $B$ by $D$ will reduce to a single solution $D = 4$ before expansion of the sub-problem $A + B = D$.*

This partial detection of independent sub-problems, which can be seen as a complex form of symmetry identification, can once again be done syntactically at the initial phase, and can be stated as follows: replace $\sum_i k_i * A_i$ by a single variable $A$ if all the $A_i$ occur only in the context of this sum i.e. in our Petri net all predecessors of $A_i$ are connected to $A_i$ with $k_i$ edges and to all other $A_j$ with $k_j$ edges and same for successors.

### 4.1.4 Experimental Results

In this section, we evaluate the two programs for enumerating minimal T-invariants and minimal P-invariants on all Biomodels.net and all Petriweb repositories already presented in the section 2.3.

#### Minimal T-invariants enumeration

On the 404 models of Biomodels.net:

- 6 models include non-integer stoichiometry and are not concerned by our approach,
- 22 models are difficult and no answer is given in a reasonable time of 10 minutes,
- 376 models: all minimal T-invariants of each model are enumerated in less than 10 seconds.

On the 80 models of Petriweb: for each model, the set of minimal T-invariants is computed in less than 1 second

#### Minimal P-invariants enumeration

On the 404 models of Biomodels.net, none required more than 1 second to compute all its minimal P-invariants.

The 80 models of Petriweb were also tested and none of them required more than one second to compute all its minimal P-invariants.

These computational results show that the CSP(FD) algorithm is very efficient for minimal P-invariants enumeration and less efficient for minimal T-invariants enumeration. We expect that this is due to the large number of minimal T-invariants

compared to the number of minimal P-invariants. It would be worthwhile to evaluate minimal T-invariants enumeration using other approaches and in particular Fourier-Motzkin based-methods. It would be interesting to see if our hard models remain so using Fourier-Motzkin algorithm. Moreover, if the number of invariants is the problem, the presented CSP approach can be easily accommodated for a partial enumeration [27].

## 4.2 Steady-state solution of biochemical systems, beyond S-Systems via T-invariants

### 4.2.1 Biochemical Systems Theory

Let us consider a biochemical system formed by a set of chemical reactions $R = R_1, R_2, ..., R_m$ and a set of chemical species $X = \{X_1, X_2, .., X_n\}$ that take part in the reactions (set of reactants union the set of products). The $j^{th}$ chemical reaction of the biochemical network is denoted as follows:

$$R_j\colon \sum_{i=1}^{n} S_{ij} X_i \Rightarrow \sum_{i=1}^{n} S'_{ij} X_i$$

where $S_{ij}$ and $S'_{ij}$ are real numbers.

In the late 1960s, Savageau introduced the Biochemical System Theory (BST) [90, 91] as a framework for modeling biochemical systems with ODEs.

The major advantage of this formalism is that it enables the modeller to describe the dynamics of a biochemical system knowing only the identity of reactants and their interconnections [102].

These ODEs have a canonical form using a power-law representation based on the General Mass Action (GMA) hypothesis. In the GMA form, rate functions are formulated as:

$$e_j = k_j \times \prod_i X_i^{S_{ij}}$$

where $X_i$ is the concentration of the $i^{th}$ species and $k_j$ is the kinetic rate constant of the $j^{th}$ reaction. This corresponds to the well-known and quite standard Mass Action kinetics derived from the Chemical Master Equation and representing intuitively the fact that reactions have a rate proportional to the amount of their reactants.

The change in the quantity of $X_i$ is thus:

$$\begin{aligned}\frac{dX_i}{dt} &= \sum_{j=1}^{n} (S'_{ij} - S_{ij}) e_j \\ &= \sum_{j=1}^{n} S'_{ij} k_j \prod_i X_i^{S_{ij}} - \sum_{j=1}^{n} S_{ij} k_j \prod_i X_i^{S_{ij}}\end{aligned}$$

i.e., variation of a species per time is a difference between two sums of power-law functions, one associated to its production and the other to its consumption.

One special case of GMA is when every species is produced by at most one reaction and consumed through at most one reaction or when reactions producing each species are dependent and reactions consuming each species are dependent so that it becomes possible to combine the sum of power laws referring to the production term into a unique power law and combine the sum referring to the consumption term into one single power law.

Even when direct combination is not possible, aggregation through an approximation can be done, for instance close to some steady states. In that case the exponents in the power law become real numbers. Note that the applicability of such an approximation for highly non-linear systems when the steady states are yet to be determined is not always clear, which is one of the motivations for this method.

Systems under this form are called S-systems where S refers to synergistic nature of this non-linear form. In S-systems, the time rate of change of a species is written as:

$$\frac{dX_i}{dt} = k_i^+ \prod_{i=1}^{n} X_i^{\alpha_i} - k_i^- \prod_{i=1}^{n} X_i^{\beta_i}$$

where $k_i^+$ and $k_i^-$ are positive rate constants. The exponential parameters $\alpha_i$ and $\beta_i$ are real.

One of the main properties of S-systems is that they can be analytically solved for steady states. Indeed being at steady state amounts to the fact that:

$$\forall i \quad \frac{dX_i}{dt} = 0 \tag{4.4}$$

which is equivalent to

$$\forall i \quad k_i^+ \prod_{i=1}^{n} X_i^{\alpha_i} = k_i^- \prod_{i=1}^{n} X_i^{\beta_i} \tag{4.5}$$

The system constituted of all equations (4.5) can be linearized by applying the logarithm function [91] and we obtain:

$$\forall i \quad \sum_{i=1}^{n} (\alpha_i - \beta_i) log(X_i) = log(\frac{k_i^-}{k_i^+}) \tag{4.6}$$

The system (4.6) can then be solved via standard Gaussian elimination.

This symbolic solution for steady states distinguishes S-systems from other non-linear ODE systems for which there is, in general, no simple solution.

In Section 4.2.2 we will explain how to generalize this search for steady-state solution using T-invariants and apply it to Biomodels.net in section 4.2.3.

Let $I$ the incidence matrix of the Petri net. We denote by $\mathcal{T}$ the (infinite) set of T-invariants. Incidence matrix and T-invariant are defined in Section 1.1.

It is important to note that T-invariants are of course already related to steadyness. However, the cone of $\mathcal{T}$ defines the steady *fluxes* of the system, but does not relate to *states*, as defined by concentrations of compounds. Depending on the kinetic laws, some fluxes might be generated by some state while some others might not. For instance, if a T-invariant requires reaction 1 to fire twice as much as reaction 2 and both have the same kinetic expression, the flux does not correspond to any state. We focus on steady states, as is classical in dynamical systems theory, and hence need to go one step further than T-invariants.

### 4.2.2 Method

Let us consider again a biochemical system formed by $R = \{R_1, R_2, ..., R_m\}$ chemical reactions and a set of chemical species $X = \{X_1, X_2, .., X_n\}$ that take part in the reactions. The $j^{th}$ chemical reaction of the biochemical network is denoted as follows:

$$R_j\colon \sum_{i=1}^{n} S_{ij} X_i \Rightarrow \sum_{i=1}^{n} S'_{ij} X_i$$

where $S_{ij}$ and $S'_{ij}$ are positive integers.

Given a chemical reaction $R_j$, we denote the set of reactant species and set of product species by $LHS(R_j) := \{X_i \in X \mid S_{ij} > 0\}$ and $RHS(R_j) := \{X_i \in X \mid S'_{ij} > 0\}$, respectively. The stoichiometric matrix $N$ is such that $N = (n_{ij}) = (S'_{ij} - S_{ij})$.

**Definition 52** (multiplicative rate laws)**.** *A rate law $e_j$ of the reaction $R_j$ is multiplicative when $e_j(X) = 0 \Leftrightarrow \exists i, S_{ij} > 0, X_i = 0$.*

Multiplicative rate laws is not very restrictive condition: GMA systems verify it but also Michaelian or Hill kinetics. In this method, it is required in order to reason structurally on the fluxes of the system. It is quite common for structural or symbolic analyses (see for instance [39, 34, 57]).Note that it is strictly more general than GMA, which is often already a requirement for any stochastic simulation/analysis.

**Definition 53** (steady state)**.** *Given a biochemical system with multiplicative rate laws $e_j$. Finding a steady state amounts to solving:*

$$\forall i \quad \frac{dX_i}{dt} = \sum_{j=1}^{m} (S'_{ij} - S_{ij}) e_j = 0$$

The system defined in 53 is — in general — non-linear and cannot be analytically solved. Nevertheless, it is possible to try and solve systems corresponding to restricted cases using T-invariants, which will result in a correct but not complete method to obtain steady states:

**Proposition 13.** *Given a biochemical system with n species and m reactions. Let* $X = (X_1, X_2, ..., X_n)$ *and* $E(X) = (e_1, e_2, ..., e_m)$ *be the vectors of species concentrations and rate laws respectively. X defines a steady state (i.e. for all* $i \in \{1, .., n\}$, $\frac{dX_i}{dt} = 0$ *) when* $E(X)$ *is a T-invariant.*

*Proof.* Considering the incidence matrix $I$ of the Petri net corresponding to the biochemical model, the system defining a steady state is equivalent to:

$$I \cdot E(X) = 0$$

where $E$ is the vector $(e_1, \ldots, e_m)$. As mentioned in Section 1.1, T-invariants are mathematically defined as positive vectors which are solutions of the equation $I \cdot V = 0$. Therefore, each steady state $X$ defines an $E(X)$ which is a T-invariant. Conversely we have:

$$E(X) \in \mathcal{T} \Rightarrow \frac{dX}{dt} = 0$$

□

Finding the $X$ solutions of the above equation is still intractable if one looks for any T-invariant, however one can state this sufficient condition in a different way:

Let $V$ be a given T-invariant. Then solving the system $E(X) = V$ will lead to steady states of the original system.

### Minimal T-invariants

Let us first consider a restricted case where:

1. $V = \alpha V', V' \in \mathcal{T}_{min}, \alpha \in \mathbb{R}_0^+$, i.e., $V$ is proportional to a minimal T-invariant;
2. $V$ (equivalently $V'$) has a GMA support, i.e.

$$\forall j \in support(V) \quad e_j = k_j \prod_{i=1}^{n} X_i^{S_{ij}}$$

Note that we only consider reaction networks with multiplicative kinetics, and that here we add the restriction that the kinetic laws of reactions in the support of the chosen T-invariant are in GMA form, thereby excluding Michaelian or other multiplicative kinetics from appearing in the support of $V$.

Solving $E(X) = V$ now amounts to solving (for $X$ and $\alpha$):

$$\begin{cases} (a) & k_j \prod_{i=1}^{n} X_i^{S_{ij}} = \alpha V'_j \quad j \in support(V') \\ (b) & \exists i, S_{ij} > 0, X_i = 0 \quad\quad j \notin support(V') \end{cases} \tag{4.7}$$

Part $(a)$ of system (4.7) is a direct consequence of our restricted setting, part $(b)$ is a consequence of the *multiplicative* nature of the kinetics.

We will explain in the next section how to solve the second part, but let us first concentrate on the first part.

Even if we did not restrict ourselves to S-systems, it is now possible to log-linearize this subsystem:

$$\sum_{i=1}^{n} S_{ij} log(X_i) - log(\alpha) = log(\frac{V_j}{k_j}) \qquad j \in support(V') \tag{4.8}$$

We obtain $|support(V')|$ *linear* equations over at most $n+1$ unknowns and can then apply Gaussian elimination or any other equivalent method to obtain a (log-)vector space of steady states. Remark that it is almost immediate to deduce the matrix corresponding to system (4.8) from $I^-$ the reactant part of the incidence matrix.

### Solving part $(b)$ of system (4.7)

Solving part $(b)$ of system (4.7) is actually a simple matter of enumeration: one tries to nullify some reactions' rate by nullifying some concentrations. However one must also verify that all $X_i$ involved in part $(a)$ are strictly positive (otherwise there is no solution).

We implemented this as a simple enumerative search in Prolog. Even for the biggest systems we tried the search for all solutions is under ten milliseconds. The only computationally expensive part being the search for T-invariants.

Remark that one could relax the restriction to multiplicative kinetics if one provides a generic way to solve $(b)$ while ensuring the feasibility of $(a)$. For instance one could consider that if a reaction has kinetic expression $e = k * (A + B) * C$ then $e = 0 \Leftrightarrow C = 0 \vee A = B = 0$. This kind of condition would be easy to incorporate into our scheme but needs to be formulated in a general way.

**Example 24.** *Let us consider again the bacteriophage T7 model of Example 15, all reactions follow the Mass Action law:*

```
MA(c1)  for          gen => tem.
MA(c2)  for          tem => _.
MA(c3)  for          tem => tem + gen.
MA(c4)  for  gen + struc => virus.
MA(c5)  for          tem => tem + struc.
MA(c6)  for        struc => _.
```

The system has 2 semi-positive minimal T-invariants: [t1, t2, t3], [t5, t6], but solving for zeroes immediately leads to the fact that the only steady state is gen = tem = struc = 0.

### Other T-invariants

The first version described above already works reasonably well at finding some steady states, however one of its shortcomings is that it only examines minimal T-invariants one by one, i.e., it restricts its search to the edges of the cone of steady fluxes.

The idea is to generalize the method while ensuring, when possible, that (4.7) remains solvable quite easily.

### Disjoint support

Let us suppose that different minimal T-invariants $V'_1, \ldots, V'_k$ have disjoint GMA supports. One can now obtain a more general version of the method introduced in Section 4.2.2:

1. $V = \sum \alpha_i V'_i, V'_i \in \mathcal{T}_{min}, \alpha \in \mathbb{R}_0^+$;
2. $\forall i \neq j \ support(V'_i) \cap support(V'_j) = \emptyset$;
3. $\forall j \in support(V) \quad e_j = k_j \prod_{i=1}^n X_i^{S_{ij}}$

Solving $E(X) = V$ now amounts to solving (for $X$ and $\alpha$):

$$\begin{cases} (a) & k_j \prod_{i=1}^n X_i^{S_{ij}} = \alpha_{j_0} V'_{j_0} & j \in support(V_{j_0}) \\ (b) & \exists i, S_{ij} > 0, X_i = 0 & j \notin support(V) \end{cases} \tag{4.9}$$

Since the supports are disjoint, there exists one and only one $j_0$ in part $(a)$ of the system (4.7).

Now, the new system is also log-linear and can thus be solved as before.

### Closed support

It is possible to obtain an even more generalized version where more combinations of T-invariants are tested by trying to solve systems like (4.9) for some arbitrary minimal T-invariant combinations.

In the general case, the (a) part will become intractable (there is a sum at the right of the "=" sign); however it can sometimes be simplified, for instance when the obtained equation is actually linear (the $S_{ij}$ on the left are equal to 0 or 1).

Since the computational cost of trying to solve the system (and stop if the symbolic computation fails) is very low compared to T-invariant computation, many combinations can easily be tried. Trying all combinations remains however definitely impossible and to guide the search an idea is to notice that, in order to solve part (b) of the system while maintaining the solvability of (a), we look for a

T-invariant combination that is *closed*, in the sense of the Chemical Organization Theory (COT) [30].

Given a set of chemical species $A \subseteq X$, its set of reaction rules is $R_A = \{\rho \in R \mid LHS(\rho) \subseteq A\}$, and the corresponding stoichiometric matrix is $N_A$. $A$ is an *organisation* if it is closed and self-maintaining. we say that $A$ is *closed* if for all reaction rules $\rho \in R_A$, $RHS(\rho) \in A$. Thus, $A$ is closed if there is no reaction with educts from $A$ producing a species not in $A$. $A$ is self-maintaining if there exists a strictly positive flux vector $v' \in \mathbb{R}_{>0}^{|R_A|}$ such that all species in $A$ are produced at a non-negative rate, that is, $N_A.v' \geq 0$ [30].

Note, nevertheless, that it is not enough to look for T-invariant combinations whose support is a minimal organization, as in [12]: we should try all organizations, and might even prefer the biggest ones. Indeed, as is the case for T-invariants, restricting the search to minimal organizations will only lead to small subproblems when the complete system might be solvable. The objective should thus be to look for T-invariant combinations such that the combined support is an organization, is as big as possible, but such that the (a) part remains solvable.

The search procedure is thus specified in the following algorithm:

**Algorithm 3** Steady state computation

Look for minimal T-invariants;
For minimal T-invariant combinations (starting with 0 or 1 invariant), try to solve the system 4.7
**if** ($b$) is not possible **then**
  add another minimal T-invariant such that it might close the support.
**else**
  try to add all other minimal T-invariants and stop if ($a$) is impossible
**end if**

**Example 25.** *Consider again the bacteriophage T7 model of Example 15 but now let us forget about the virus, which will otherwise always increase; one obtains:*

```
MA(c1)   for          gen => tem.
MA(c2)   for          tem => _.
MA(c3)   for          tem => tem + gen.
MA(c4)   for  gen + struc => _.
MA(c5)   for          tem => tem + struc.
MA(c6)   for       struc => _.
```

rule 4 no longer includes the virus

*Now the system gets a third minimal T-invariant: [t1, t2, t3], [t3, t4, t5], [t5, t6], but most importantly another steady state is found (it is the only non-trivial one actually), using a combination of all the minimal invariants (even if the supports*

*are not disjoint):*

$$\begin{aligned} tem &= \frac{c1 * c6 * (c2 - c3)}{c2 * c4 * (c3 - c2 - c5)} \\ struc &= \frac{c1 * (c3 - c2)}{c2 * c4} \\ gen &= \frac{c6 * (c2 - c3)}{c4 * (c3 - c2 - c5)} \end{aligned}$$

We remark that other combinations of T-invariants, like MCT-sets [89], might be used, however they will still need to allow to solve (a) and (b), and thus to have as support an organization, hence the choice to remain at that level for now.

### 4.2.3 Results

In this section, we apply the proposed method to the models of the Biomodels.net repository[2]. Among the 241 curated models, 14 do not include any continuous part (ODEs) and are thus not targeted by our method, for the other ones the structure and kinetic laws were extracted from the SBML by our tool, ignoring any other information (like events).

Using the Nicotine tool implementing the techniques described above, some steady states were found for 94 models out of the remaining 227, with a time-out of two minutes. Among the remaining 133 models, for which Nicotine could not find any solution, only 31 hit the time-out of two minutes. In most of the cases, the tested combinations of T-invariants do not lead to any steady state computation because of the strict conditions on the applicability of this method (namely multiplicative kinetics and GMA support, as explained in Section 4.2.2). As we will show below it is often possible to restructure an SBML model such that these conditions are met.

This problem reflects the fact that modellers often tend to combine several separate reactions into one, leading to a wrong structure (and non-multiplicative kinetics). For instance, in model 149 of ERK crosstalk, the "Axin synthesis" reaction R14 is given as a single reaction X11 + X14 => X11 + X14 + X12, i.e., a synthesis of X12 with two modifiers, X11 and X14 and a single rate law: $v_{14} = k_{14} + k_{21} * (\mathrm{X11} + \mathrm{X14})$.

Splitting that reaction into:

```
MA(k14)   for   _      =>      X12.
MA(k21)   for   _   =[X11]=>   X12.
MA(k21)   for   _   =[X14]=>   X12.
```

would have allowed a multiplicative kinetics with the same ODEs but a differently structured model.

[2]dated January 2010

It is worth notifying that trying to solve analytically all the models, even with state-of-the-art symbolic computation software like Maple 13, does not solve all problems either. Indeed, of the 227 curated SBML models with an ODE part, 105 models do not provide any solution when imported into Maple for steady-state search, with a time-out of two minutes.

Interestingly, Nicotine finds some steady states for 31 models of the 105 Maple failures to give any steady-state solution even though there are ODEs.

Consider for instance model 46, describing the mechanism of protection of peroxidase activity by oscillatory dynamics It contains 16 places (chemical species) and 15 transitions (chemical reactions). All kinetics laws are multiplicative, the method of Section 4.2.2 thus applies and we obtain some steady states in 28ms[3], even if Maple did not provide any result.

The minimal T-invariant [v131, v132] gives four families of steady states represented in the following table, and corresponding to the different ways to solve for zeroes:

| species | family1 | family2 | family3 | family4 |
|---|---|---|---|---|
| NADH | 0 | 0 | 0 | 0 |
| O2 | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| H2O2 | * | * | 0 | 0 |
| per3 | 0 | 0 | * | * |
| coI | * | 0 | * | 0 |
| ArH | 0 | * | 0 | * |
| coII | * | 0 | * | 0 |
| Ar | * | * | * | * |
| NADrad | 0 | 0 | 0 | 0 |
| super | 0 | 0 | 0 | 0 |
| coIII | * | * | * | * |
| per2 | 0 | 0 | 0 | 0 |
| NAD2 | * | * | * | * |
| NAD | * | * | * | * |
| O2g | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| NADHres | 0 | 0 | 0 | 0 |

In this table the symbol (*) denotes that any concentration value (positive or null) is acceptable. It shows that, to the considered minimal T-invariant corresponds an infinity of steady states where some elements may have non zero concentration values.

For the non-null values, Gaussian elimination solves the system of linear equations. In this example, an equilibrium is established between O2 and O2g:

$$\frac{\mathrm{O2}}{\mathrm{O2g}} = \frac{k13f}{k13b}$$

[3]computation time on a PC with an intel Core2 Quad processor 2.8GHz and 8Go of memory. We used the same PC for the whole procedure, including T-invariant computation

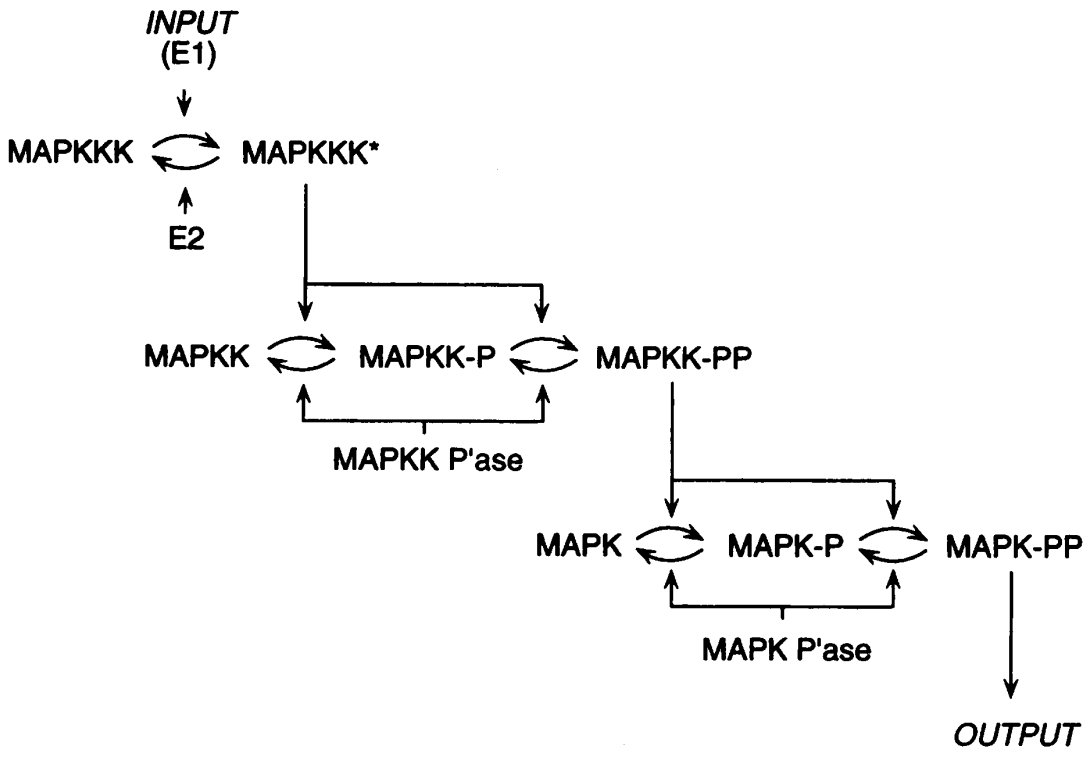



Figure 4.2: A diagram describing the model 9 of the Biomodels.net repository.

where $k13f$ and $k13b$ are respectively the rate constants of reactions v131 and v132.

Some steady states can be extracted even from the trivial T-invariant: 8 families of concentrations are computed, in all of them the concentrations of O2, NADrad, super, O2g and NADres is null and the concentration of coIII, NAD2 and NAD have positive or null values.

Some more complex steady states can also be found as is the case for model number 9 of the Biomodels.net database, describing the classical Huang and Ferrell model of the MAPK cascade [53] (see Figure 4.2). Note that since the structure of this model was used as a basis for [67], it has already been studied in detail in [41].

The figure 4.2 corresponds to Figure 1 of [53]. Note that in the SBML model and thus in our results, the names are changed from "MAPK-P" to "P_K" (i.e., drop "MAP" and put phosphorylations first).

This model contains 25 places and 30 transitions; steady states computation is fulfilled in 220 ms and reveals 15 minimal T-invariants.

The family of T-invariants [[r1a, r1arev], [r2a, r2arev], [r2a, r2b, r1a, r1b]], which denotes a linear combination of the three minimal T-invariants [r1a, r1arev], [r2a, r2arev] and [r2a,r2b,r1a,r1b], gives four families of steady states represented in the following table:

| species | family1 | family2 | family3 | family4 |
|---|---|---|---|---|
| E1 | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| E2 | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| KKK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| P_KKK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| KK | 0 | 0 | 0 | 0 |
| P_KK | 0 | 0 | 0 | 0 |
| PP_KK | 0 | 0 | * | * |
| K | * | * | 0 | 0 |
| P_K | 0 | * | 0 | 0 |
| PP_K | 0 | * | 0 | * |
| KPase | * | 0 | * | 0 |
| KKPase | * | * | 0 | 0 |
| E1_KKK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| E2_P_KKK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| P_KKK_KK | 0 | 0 | 0 | 0 |
| P_KKK_P_KK | 0 | 0 | 0 | 0 |
| PP_KK_K | 0 | 0 | 0 | 0 |
| PP_KK_P_K | 0 | 0 | 0 | 0 |
| KKPase_PP_KK | 0 | 0 | 0 | 0 |
| KKPase_P_KK | 0 | 0 | 0 | 0 |
| KPase_PP_K | 0 | 0 | 0 | 0 |
| KPase_P_K | 0 | 0 | 0 | 0 |
| K_PP_norm | * | * | * | * |
| KK_PP_norm | * | * | * | * |
| KKK_P_norm | * | * | * | * |

For each of these families, we obtain a steady state when the non null values in the table, E1, E2, KKK, P_KKK, E1_KKK and E2_P_KKK satisfy the following equations:

$$\begin{aligned} \frac{\text{E1} * \text{KKK}}{\text{E1\_KKK}} &= \frac{k2 + d1}{a1} \\ \text{E1\_KKK} &= \text{E2\_P\_KKK} \\ \frac{\text{E2} * \text{P\_KKK}}{\text{E2\_P\_KKK}} &= \frac{d2 + k2}{a2} \end{aligned}$$

where $k2, d1, a1$ and $a2$ are the rate constants of r1b (and r2b), r1arev, r1a and r2a respectively.

This can be summed up by stating that if most of the complexes involved in the cascade are absent, most notably those involving phosphatases and the second-level kinase, there exists a steady state where the first and third level maintain some equilibrium.

A second set of steady states results from the T-invariant combination [[r3a, r3arev], [r4a, r4arev], [r4a, r4b, r3a, r3b], [r5a, r5arev], [r6a, r6arev], [r6a, r6b, r5a, r5b]] and defines again four families of steady states as follows:

| species | family1 | family2 | family3 | family4 |
|---|---|---|---|---|
| E1 | $*$ | $*$ | 0 | 0 |
| E2 | 0 | 0 | 0 | 0 |
| KKK | 0 | 0 | $*$ | $*$ |
| P_KKK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| KK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| P_KK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| PP_KK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| K | 0 | 0 | 0 | 0 |
| P_K | 0 | 0 | 0 | 0 |
| PP_K | 0 | $*$ | 0 | $*$ |
| KPase | $*$ | 0 | $*$ | 0 |
| KKPase | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| E1_KKK | 0 | 0 | 0 | 0 |
| E2_P_KKK | 0 | 0 | 0 | 0 |
| P_KKK_KK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| P_KKK_P_KK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| PP_KK_K | 0 | 0 | 0 | 0 |
| PP_KK_P_K | 0 | 0 | 0 | 0 |
| KKPase_PP_KK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| KKPase_P_KK | $*>0$ | $*>0$ | $*>0$ | $*>0$ |
| KPase_PP_K | 0 | 0 | 0 | 0 |
| KPase_P_K | 0 | 0 | 0 | 0 |
| K_PP_norm | $*$ | $*$ | $*$ | $*$ |
| KK_PP_norm | $*$ | $*$ | $*$ | $*$ |
| KKK_P_norm | $*$ | $*$ | $*$ | $*$ |

The additional equations are:

$$\begin{aligned}
\frac{\mathrm{KK} * \mathrm{P_KKK}}{\mathrm{P_KKK_KK}} &= \frac{k3+d3}{a3} \\
\frac{\mathrm{KKPase} * \mathrm{P_KK}}{\mathrm{KKPase_P_KK}} &= \frac{d4+k4}{a4} \\
\frac{\mathrm{KKPase_PP_KK}}{\mathrm{P_KK} * \mathrm{P_KKK}} &= \frac{a5*k5}{k6*(d5+k5)} \\
\frac{\mathrm{KKPase_P_KK}}{\mathrm{P_KKK_KK}} &= \frac{k3}{k4} \\
\frac{\mathrm{P_KK} * \mathrm{P_KKK}}{\mathrm{P_KKK_P_KK}} &= \frac{k5+d5}{a5} \\
\frac{\mathrm{PP_KK} * \mathrm{P_KKK_KK}}{P_KK^2 * P_KKK} &= \frac{a5*k5*(d6+k6)*a4*k4}{k3*k6*(d4+k4)*a6*(k5+d5)}
\end{aligned}$$

This complex equilibrium (dimension 9, ignoring the variables completely free in the above table) describes a steady state where the last level is off (K = 0) but the first two are actually active, especially the intermediary level.

In the same way, steady states are found for the T-invariant combination [[r10a, r10arev], [r10a, r10b, r9a, r9b], [r7a, r7arev], [r8a, r8arev], [r8a, r8b, r7a, r7b], [r9a, r9arev]]:

| species | family1 | family2 | family3 | family4 |
|---|---|---|---|---|
| E1 | $*$ | $*$ | 0 | 0 |
| E2 | $*$ | 0 | $*$ | 0 |
| KKK | 0 | 0 | $*$ | $*$ |
| P_KKK | 0 | $*$ | 0 | $*$ |
| KK | $*$ | 0 | $*$ | 0 |
| P_KK | $*$ | 0 | $*$ | 0 |
| PP_KK | $* > 0$ | $* > 0$ | $* > 0$ | $* > 0$ |
| K | $* > 0$ | $* > 0$ | $* > 0$ | $* > 0$ |
| P_K | $* > 0$ | $* > 0$ | $* > 0$ | $* > 0$ |
| PP_K | $* > 0$ | $* > 0$ | $* > 0$ | $* > 0$ |
| KPase | $* > 0$ | $* > 0$ | $* > 0$ | $* > 0$ |
| KKPase | 0 | 0 | 0 | 0 |
| E1_KKK | 0 | 0 | 0 | 0 |
| E2_P_KKK | 0 | 0 | 0 | 0 |
| P_KKK_KK | 0 | 0 | 0 | 0 |
| P_KKK_P_KK | 0 | 0 | 0 | 0 |
| PP_KK_K | $* > 0$ | $* > 0$ | $* > 0$ | $* > 0$ |
| PP_KK_P_K | $* > 0$ | $* > 0$ | $* > 0$ | $* > 0$ |
| KKPase_PP_KK | 0 | 0 | 0 | 0 |
| KKPase_P_KK | 0 | 0 | 0 | 0 |
| KPase_PP_K | $* > 0$ | $* > 0$ | $* > 0$ | $* > 0$ |
| KPase_P_K | $* > 0$ | $* > 0$ | $* > 0$ | $* > 0$ |
| K_PP_norm | $*$ | $*$ | $*$ | $*$ |
| KK_PP_norm | $*$ | $*$ | $*$ | $*$ |
| KKK_P_norm | $*$ | $*$ | $*$ | $*$ |

and some complex equilibrium between PP_KK, K, P_K, PP_K, KPase, PP_KK_K , PP_KK_P_K, KPase_PP_K, and KPase_P_K is defined by the following equations:

$$\begin{aligned}
\frac{\mathrm{K} * \mathrm{PP_KK}}{\mathrm{PP_KK_K}} &= \frac{k7 + d7}{a7} \\
\frac{\mathrm{KPase} * \mathrm{P_K}}{\mathrm{KPase_P_K}} &= \frac{d8 + k8}{a8} \\
\frac{\mathrm{KPase_PP_K}}{\mathrm{PP_KK_P_K}} &= \frac{k9}{k10} \\
\frac{\mathrm{KPase_P_K}}{\mathrm{PP_KK_K}} &= \frac{k7}{k8} \\
\frac{\mathrm{PP_K} * \mathrm{PP_KK_K}}{\mathrm{PP_KK_P_K} * \mathrm{P_K}} &= \frac{k8 * k9 * a8 * (k10 + d10)}{k7 * k10 * (d8 + k8) * a10} \\
\frac{\mathrm{PP_KK} * \mathrm{P_K}}{\mathrm{PP_KK_P_K}} &= \frac{d9 + k9}{a9}
\end{aligned}$$

This shows that the two last levels can also remain in a complex equilibrium.

Finally, we remark that for this model too the trivial T-invariant leads to some steady states.

### 4.2.4 Conclusions and Perspectives

The first step of our method relies on the computation of the T-invariants of the model's Petri net. This, however introduces two limitations:

1. actually, most of the computation time is spent on this first stage. Indeed, T-invariant computation is a hard problem (actually quite harder in practice than P-invariant computation on biochemical systems, though they have the same theoretical complexity. This seems related to the fact that in biochemical reaction networks the degree of the transitions is usually quite smaller than that of the places).

2. the T-invariants, and even the following step of solving system (4.9) suppose that the structure of the PN is *coherent* with the kinetics. Namely we require multiplicative kinetics for the whole model (and GMA for the support of some T-invariants).

Even if T-invariant computation gets more and more efficient (see for instance [17] for some recent work using decision diagrams), it is quite crucial to note that when T-invariant computation reveals time consuming, the proposed technique can work as soon as some (minimal, or even just candidates-minimal) T-invariants are found. Two examples that come to mind are thus the use of Nicotine [94] to compute T-invariants with an increasing bound on the integer domain, or to use [27] to compute the K-shortest minimal T-invariants, before (or while) proceeding to invariant composition and solving.

Addressing the other limitation is actually a much more general question. Indeed more and more formal techniques extract qualitative information from the structure of biochemical models, however the current status of hand-written models in web-based repositories is that the structure might be quite different from what the original modellers had in mind (or on diagram), even if the models are "curated". This issue applies to model-checking, abstract interpretation from the structure [34], stochastic simulation *à la* Gillespie [42], Chemical Reaction Network Theory [39, 96], etc. Some authors have already proposed solutions to check if the kinetics and the structure were at least coherent in some sense, notably [57] in order to use COT. Proposals allowing to obtain a *properly structured* model, as was done in the beginning of Section 4.2.3 are also under way (an article has recently been submitted about this topic by the second author). If everything else fails, it remains possible, as outlined in Section 4.2.2, to provide constraints corresponding to the non-multiplicative kinetics of the model and allowing to reason on the nullity of the kinetic expression.

Another noteworthy remark about the proposed method is that it is in general incomplete, since combinations involving many T-invariants usually result in a non-log-linearizable system. However, the method can be complete under certain conditions. For instance, for both versions of Example 15, the steady states that are found are the only ones, and the method can certify this fact. In this specific case the proof is quite simple: all combinations of minimal T-invariants are tried and either lead to no solution (when solving for zeroes) or are solvable. Finding more general conditions under which the described technique ensures that all possible steady states were found is one of our current perspectives.

# General Conclusion

In this thesis, we argue that constraint programming over finite domains can be successfully applied to structural problems like invariants of places and transitions siphons/traps. These structural properties can give us some dynamical information about biochemical reaction network. Their computation can benefit from the know-how developed for finite domain constraint programming solving, like symmetry breaking, search heuristics, flexibility, etc. We used constraint programming for three topical issues: enumerating minimal siphons and traps of a Petri net, enumerating minimal invariants of a Petri net and to compute, in a fully analytical way, steady states of biochemical systems defined by a system of ODEs. In the following, we summarize the thesis content.

- In the first chapter we have defined basic notions and notations of the Petri net formalism. We have presented a new complexity result of minimal siphons extraction problem in Petri nets with bounded tree-width. We have also developed formal links between the structural properties of siphons and traps in Petri nets and their dynamical properties in Computation Tree Logic: we have shown that siphons and traps entail a family of particular stability properties that can be characterized by a fragment of Computation Tree Logic.
- In the second chapter, we have shown the use of Petri nets for the description and the analysis of biochemical reaction nets by focusing on basic structural properties regarding the dynamics of a biochemical reactions model.
- In the third chapter, we have described a boolean model for the problem of enumerating all minimal siphons in a Petri net and have compared two Boolean methods to a state-of-the-art algorithm from the Petri net community. The miniSAT solver and the CLP(B) program based on GNU-Prolog solver both solve our large benchmark of real-size problems and outperform the dedicated algorithm by two orders of magnitude. We provide benchmark from biology composed of 403 biological models in biomodels.net, and a general benchmark from Petriweb repository. The Boolean method for enumerating all minimal siphons using miniSAT is very efficient. It also scales very well in the size of the net. The surprising efficiency of the miniSAT and CLP(B) methods for solving the practical instances of this NP-complete problem has been analysed in connection to the well-known phase transition phenomenon in 3-SAT. In addition, we show that in networks with bounded tree-width, the existence of a minimal siphon containing a given set of places can be decided in linear time.

The tree-width of Petri-nets associated to even very large biochemical models, composed of several hundreds of biochemical species and reactions, remains limited to small values in practice.

These results militate for the analysis of biochemical networks with Petri net concepts and Constraint Programming tools.

- In the fourth chapter, we dealt with the problem of invariants computation and we have proposed a simple implementation based on GNU-Prolog's finite domain solver, and including symmetry detection and breaking, was incorporated into the Biocham modelling environment and in the independent tool Nicotine. We have also presented a new method to compute, in a fully analytical way, steady states of biochemical systems defined by a system of ODEs.

For each one on the three issues we dealt with, the use of constraint programming approaches turned out to be beneficial. In general, constraints allow one to easily model problems. Once a constraint model is defined, a variety of constraint solver with high performance is available. The search for solutions exploits the defined constraints to prune the solution space. The search can be improved by new properties that can be formalized as constraints that can be added to model to further reduce the solution space. As a further work, we intend to generalize our approach to other problems of this category.

To conclude, the result of this work is twofold: on the one hand, it confirms the conviction that structural properties are a key to master the complexity of biochemical reaction networks; on the other hand, it shows that Petri nets provide a unified formalism to link systems biology to different domains such as model-Checking, Constraint Programming, structural analysis, dynamical systems' steady-state analysis, S-Systems and Chemical Reaction Network Theory, abstract Interpretation, Ordinary Differential Equations, etc. These latter points confirm that formal techniques from Computer Science are promising to challenges in the analysis of biochemical reaction networks.

# Appendix A

# Appendix 1

In this appendix chapter, we provide the pseudo-code of the state-of-the-art algorithm considered for enumerating minimal siphons [23].

function $\Sigma_\Pi = SolveList(\Lambda)$
$\Sigma_\Pi = \emptyset$
**while** $\Lambda \neq ()$ **do**
  $\Pi = pop(\Lambda)$
  $(S, \Pi) = FindSiphon(\Pi)$
  **if** $S \neq \emptyset$ **then**
    **if** $S \neq P_{in}$ **then**
      $S = FindMinimalSiphon(\Pi)$
    **end if**
    $\Sigma_\Pi = \Sigma_\Pi \cup \{S\}$
    $\Lambda = ((\Pi), \Lambda)$
    $\Lambda = Partition(\Lambda, S)$
  **end if**
**end while**

function $(S, \Pi') = FindSiphon(\Pi)$
$\Pi' = \Pi$
$isReducible = true$
**while** $isReducible$ **do**
  **if** $P_{in} \cap P_{out} \neq \emptyset$ **then**
    $S = \emptyset$
    $return$
  **else**
    **if** ${}^\bullet P_{in} \cup P_{in}^\bullet = P$ **then**
      $S = P_{in}$
    **else**
      $S = \emptyset$
    **end if**
    $return$

**end if**
**if** $P_{out} \neq \emptyset$ **then**
$G = red(G, P - P_{out})$
$\Pi' = (G, P_{in}, \emptyset)$
**end if**
$(isReducible, \Pi') = Reduce(\Pi')$
**end while**
$S = P$

function $(isReducible, \Pi') = Reduce(\Pi)$
$\Pi' = \Pi$
$isReducilbe = true$
$\bar{T} = \{t \in T, \text{ such that } {}^\bullet t = \emptyset\}$, $\bar{P} = \bar{T}^\bullet$
$\hat{T} = \{t \in {}^\bullet P_{in} - P_{in}^\bullet, \text{ such that } |{}^\bullet t| = 1\}$, $\hat{P} = {}^\bullet \hat{T} \cap (P - P_{in})$
**if** $\bar{P} = \emptyset$ and $\hat{P} = \emptyset$ **then**
$isReducible = false$
**else**
$\Pi' = (G, P_{in} \cup \hat{P}, P_{out} \cup \bar{P})$
**end if**

function $S = FindMinimalSiphon(\Pi)$
$S = P$, $\tilde{P} = S - P_{in}$
**while** $\tilde{P} \neq \emptyset$ **do**
$p = Get(\tilde{P})$
**if** $\{p\} \subset ({}^\bullet t \cap S)$ or $t^\bullet \cap S = \emptyset$, $\forall t \in p^\bullet$ **then**
$S = S - \{p\}$
**end if**
$\tilde{P} = \tilde{P} - \{p\}$
**end while**
$\tilde{P} = S - P_{in}$, $\tilde{P_{in}} = P_{in}$
**while** $\tilde{P} \neq \emptyset$ **do**
$p = Get(\tilde{P})$, $\tilde{G} = red(G, S - \{p\})$, $\tilde{\Pi} = (\tilde{G}, \tilde{P_{in}}, \emptyset)$
$(\tilde{S}, \tilde{\Pi}) = FindSiphon(\tilde{\Pi})$
**if** $\tilde{(S)} \neq \emptyset$ **then**
$S = \tilde{S}$, $\tilde{P} = S - \tilde{P_{in}}$, $G = \tilde{G}$
**else**
$\tilde{P} = \tilde{P} - \{p\}$, $\tilde{P_{in}} = P_{in} \cup \{p\}$
**end if**
**end while**

function $\tilde{\Lambda} = Partition(\Lambda, S)$
$\tilde{\Lambda} = ()$
$\Pi = pop(\Lambda)$
$\tilde{P} = S - P_{in}$
**while** $\tilde{P} \neq \emptyset$ and $P_{out} \cap P_{in} = \emptyset$ **do**
$p = Get(\tilde{P})$

$\Pi = (G, P_{in}, P_{out} \cup \{P\})$
$\tilde{\Lambda} = (\tilde{\Lambda}, (\Pi))$
$\tilde{P} = \tilde{P} - \{p\}$, $P_{in} = P_{in} \cup \{p\}$
**end while**$\tilde{\Lambda} = (\tilde{\Lambda}, (\Lambda))$

Algorithm 1 and 2 differ only by partition function:

function $\tilde{\Lambda} = Partition(\Lambda, S)$
$\tilde{\Lambda} = ()$
**while** $\Lambda \neq \emptyset$ **do**
$\Pi = pop(\Lambda)$
$\tilde{P} = S - P_{in}$
**while** $\tilde{P} \neq \emptyset$ and $P_{out} \cap P_{in} = \emptyset$ **do**
$p = Get(\tilde{P})$
$\Pi = (G, P_{in}, P_{out} \cup \{P\})$
$(\tilde{S}, \Pi) = FindSiphon(\Pi)$
**if** $\tilde{S} \neq \emptyset$ **then**
$\tilde{\Lambda} = (\tilde{\Lambda}, (\Pi))$
**end if**
$\tilde{P} = \tilde{P} - \{p\}$, $P_{in} = P_{in} \cup \{p\}$
**end while**
**end while**

# Bibliography


[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell, fourth edition*. Garland Science, 2002.

[2] A. Alfonsi, E. Cancès, G. Turinici, B. di Ventura, and W. Huisinga. Adaptive simulation of hybrid stochastic and deterministic models for biochemical systems. *ESAIM: Proc.*, 14:1–13, Sept. 2005.

[3] R. Alur, T. A.Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

[4] K. R. Apt. *Principles of Constraint Programming*. Cambridge University Press, New York, NY, USA, 2003.

[5] M. R. Birtwistle, M. Hatakeyama, N. Yumoto, B. A. Ogunnaike, J. B. Hoek, and B. N. Kholodenko. Ligand-dependent responses of the ErbB signaling network: experimental and modeling analysis. *Molecular Systems Biology*, 3(144), Sept. 2007.

[6] A. Bockmayr and A. Courtois. Using hybrid concurrent constraint programming to model dynamic biological systems. In Springer-Verlag, editor, *Proceedings of ICLP'02, International Conference on Logic Programming*, volume 2401 of *Lecture Notes in Computer Science*, pages 85–99, Copenhagen, 2002.

[7] H. Bodlaender. Classes of graphs with bounded tree-width. *Bulletin of EATCS*, pages 116–128, 1988.

[8] M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing finite kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59:115–131, 1988.

[9] L. Calzone, F. Fages, and S. Soliman. BIOCHAM: An environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics*, 22(14):1805–1807, 2006.

[10] L. Calzone, A. Gelay, A. Zinovyev, F. Radvanyi, and E. Barillot. A comprehensive modular map of molecular interactions in RB/E2F pathway. *Molecular Systems Biology*, 4(173), 2008.

[11] R. Cavada, A. Cimatti, E. Olivetti, M. Pistore, and M. Roveri. *NuSMV 2.1 User Manual*. CMU and ITC-irst, IRST - Via Sommarive 18, 38055 Povo (Trento) - Italy, 1998–2002.

[12] F. Centler, C. Kaleta, P. S. di Fenizio, and P. Dittrich. Computing chemical organizations in biological networks. *Bioinformatics*, 24(14):1611–1618, May 2008.

[13] N. Chabrier and F. Fages. Symbolic model checking of biological systems. In *Poster proceedings of European Conference on Computational Biology ECCB'02*, Saarbrück, Sept. 2002.

[14] N. Chabrier and F. Fages. Symbolic model checking of biochemical networks. In C. Priami, editor, *CMSB'03: Proceedings of the first workshop on Computational Methods in Systems Biology*, volume 2602 of *Lecture Notes in Computer Science*, pages 149–162, Rovereto, Italy, Mar. 2003. Springer-Verlag.

[15] N. Chabrier-Rivier, M. Chiaverini, V. Danos, F. Fages, and V. Schächter. Modeling and querying biochemical interaction networks. *Theoretical Computer Science*, 325(1):25–44, Sept. 2004.

[16] F. Chu and X.-L. Xie. Deadlock analysis of petri nets using siphons and mathematical programming. *IEEE Transactions on Robotics and Automation*, 13(6):793–804, 1997.

[17] G. Ciardo, G. Mecham, E. Paviot-Adet, and M. Wan. P-semiflow computation with decision diagrams. In *PETRI NETS '09: Proceedings of the 30th International Conference on Applications and Theory of Petri Nets*, pages 143–162, Berlin, Heidelberg, 2009. Springer-Verlag.

[18] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[19] J. M. Colom and M. Silva. Convex geometry and semiflows in P/T nets. a comparative study of algorithms for computation of minimal p-semiflows. In *Proceedings of the 10th International Conference on Application and Theory of Petri Nets*, pages 74–95, Bonn, Germany, 1989.

[20] F. Corblin, S. Tripodi, E. Fanchon, D. Ropers, and L. Trilling. A declarative constraint-based method for analyzing discrete genetic regulatory networks. *Biosystems*, 98(2):91–104, 2009.

[21] R. Cordone, L. Ferrarini, and L. Piroddi. Characterization of minimal and basis siphons with predicate logic and binary programming. In *Proceedings of IEEE International Symposium on Computer-Aided Control System Design*, pages 193–198, 2002.

[22] R. Cordone, L. Ferrarini, and L. Piroddi. Some results on the computation of minimal siphons in petri nets. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii USA, dec 2003.

[23] R. Cordone, L. Ferrarini, and L. Piroddi. Enumeration algorithms for minimal siphons in petri nets based on place constraints. *IEEE transactions on systems, man and cybernetics. Part A, Systems and humans*, 35(6):844–854, 2005.

[24] B. Courcelle. The monadic second-order logic of graphs i. recognizable sets of finite graphs. *Information and Computation*, pages 12–75, 1990.

[25] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL'77: Proceedings of the 6th ACM Symposium on Principles of Programming Languages*, pages 238–252, New York, 1977. ACM Press. Los Angeles.

[26] J. M. Crawford and L. D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings of the 11th National Conference on Artificial Intelligence*, pages 21–27. AAAI press, 1993.

[27] L. F. de Figueiredo, A. Podhorski, A. Rubio, C. Kaleta, J. E. Beasley, S. Schuster, and F. J. Planes. Computing the shortest elementary flux modes in genome-scale metabolic networks. *Bioinformatics*, 25(23):3158–3165, Dec. 2009.

[28] V. Devloo, P. Hansen, and M. Labbe. Identification of all steady states in large biological systems by logical analysis. *Bulletin of Mathematical Biology*, 65:1025–1051, 2003.

[29] D. Diaz and P. Codognet. Design and implementation of the GNU Prolog system. *Journal of Functional and Logic Programming*, 6, Oct. 2001.

[30] P. Dittrich and P. di Fenizio. Chemical organisation theory. *Bulletin of Mathematical Biology*, 69(4):1199–1231, Apr. 2007.

[31] A. Doi, S. Fujita, H. Matsuno, M. Nagasaki, and S. Miyano. Construction biological pathway models with hybrid functional petri nets. *In Silico Biology*, 4:271–291, 2004.

[32] N. Dudani, S. Ray, S. George, and U. Bhalla. Multiscale modeling and interoperability in MOOSE. *BMC Neuroscience*, 10(Suppl 1):P54, 2009.

[33] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and M. K. Sönmez. Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the seventh Pacific Symposium on Biocomputing*, pages 400–412, Jan. 2002.

[34] F. Fages and S. Soliman. Abstract interpretation and types for systems biology. *Theoretical Computer Science*, 403(1):52–70, 2008.

[35] F. Fages, S. Soliman, and N. Chabrier-Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4(2):64–73, Oct. 2004.

[36] F. Fages, S. Soliman, and R. Coolen. CLPGUI: a generic graphical user interface for constraint logic programming. *Journal of Constraints, Special Issue on User-Interaction in Constraint Satisfaction*, 9(4):241–262, Oct. 2004.

[37] C. Fall, E. Marland, J. Wagner, and J. Tyson. *Computational Cell Biology*. Springer, 2002.

[38] E. Fanchon, F. Corblin, L. Trilling, B. Hermant, , and D. Gulino. Modeling the molecular network controlling adhesion between human endothelial cells: Inference and simulation using constraint logic programming. In *CMSB'04: Proceedings of the 20 international conference on Computational Methods in Systems Biology*, pages 104–118. Springer-Verlag, 2004.

[39] M. Feinberg. Mathematical aspects of mass action kinetics. In L. Lapidus and N. R. Amundson, editors, *Chemical Reactor Theory: A Review*, chapter 1, pages 1–78. Prentice-Hall, 1977.

[40] E. C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proceedings of AAAI'91*, pages 227–233. MIT Press, 1991.

[41] D. Gilbert, M. Heiner, and S. Lehrack. A unifying framework for modelling and analysing biochemical pathways using petri nets. In M. Calder and S. Gilmore, editors, *CMSB'07: Proceedings of the fifth international conference on Computational Methods in Systems Biology*, volume 4695 of *Lecture Notes in Computer Science*, Edinburgh, Scotland, Sept. 2007. Springer-Verlag.

[42] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[43] P. Goss and J.Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets. *Proceedings of the National Academy of the United States of America*, 95(12):6750–6755, June 1998.

[44] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural csp decomposition methods. *Artificial Intelligence*, 124:2000, 2000.

[45] R. Goud, K. van Hee, R. Post, and J. van der Werf. Petriweb: A repository for petri nets. In S. Donatelli and P. Thiagarajan, editors, *Petri Nets and Other Models of Concurrency - ICATPN 2006*, volume 4024 of *Lecture Notes in Computer Science*, pages 411–420. Springer-Verlag, 2006.

[46] S. Grunwalda, A. Speera, J. Ackermanna, and I. Kocha. Petri net modelling of gene regulation of the duchenne muscular dystrophy. *Biosystems*, 92(2):189–205, May 2008.

[47] M. Heiner, D. Gilbert, and R. Donaldson. Petri nets for systems and synthetic biology. In M. Bernardo, P. Degano, and G. Zavattaro, editors, *8th Int. School on Formal Methods for the Design of Computer, Communication and Software Systems: Computational Systems Biology SFM'08*, volume 5016 of *Lecture Notes in Computer Science*, pages 215–264, Bertinoro, Italy, Feb. 2008. Springer-Verlag.

[48] M. Heiner, C. Mahulea, and M. Silva. On the importance of the deadlock trap property for monotonic liveness. In *Int. Workshop on Biological Processes and Petri Nets (BioPPN), A satellite event of Petri Nets 2010*, 2010.

[49] S. Helfert, A. Estevez, B. Bakker, P. Michels, and C. Clayton. Roles of triosephosphate isomerase and aerobic metabolism in trypanosoma brucei. *Biochem. J.*, 357:117–125, 2001.

[50] R. Hofestädt. A petri net application to model metabolic processes. *Systems Analysis Modelling Simulation*, 16:113–122, Oct. 1994.

[51] R. Hofestädt and S. Thelen. Quantitative modeling of biochemical networks. In *In Silico Biology*, volume 1, pages 39–53. IOS Press, 1998.

[52] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. Copasi – a complex pathway simulator. *Bioinformatics*, 22(24):3067–3074, 2006.

[53] C.-Y. Huang and J. E. Ferrell, Jr. Ultrasensitivity in the mitogen-activated protein kinase cascade. *PNAS*, 93(19):10078–10083, Sept. 1996.

[54] M. Hucka et al. The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[55] K. Jensen. Coloured petri nets and the invariant-method. *Theoretical Computer Science*, 14:317–336, 1981.

[56] K. Jensen, L. M. Kristensen, and L. Wells. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 9:213–254, 2007.

[57] C. Kaleta, S. Richter, and P. Dittrich. Using chemical organization theory for model checking. *Bioinformatics*, 25(15):1915–1922, 2009.

[58] M. Kinuyama and T. Murata. Generating siphons and traps by petri net representation of logic equations. In *Proceedings of 2th Conference of the Net Theory SIG-IECE*, pages 93–100, 1986.

[59] H. Kitano. Systems biology: A brief overview. *Science*, 295(5560):1662–1664, Mar. 2002.

[60] I. Koch, B. Junker, and M. Heiner. Application of petri net theory for modelling and validation of the sucrose breakdown pathway in the potato tuber. *Bioinformatics*, 21:1219–1226, Apr. 2005.

[61] K. W. Kohn. Molecular interaction map of the mammalian cell cycle control and DNA repair systems. *Molecular Biology of the Cell*, 10(8):2703–2734, Aug. 1999.

[62] E. Korach and N. Solel. Tree-width, path-width, and cutwidth. *Discrete Applied Mathematics*, 43(1):97–101, May 1993.

[63] A. Larhlimi and A. Bockmayr. A new constraint-based description of the steady-state flux cone of metabolic networks. *Discrete Applied Mathematics*, 157(10):2257–2266, 2009. Networks in Computational Biology.

[64] K. Lautenbach. Linear algebraic calculation of deadlocks and traps. In G. Voss and Rozenberg, editors, *Concurrency and Nets Advances in Petri Nets*, pages 315–336, New York, 1987. Springer-Verlag.

[65] C. F. Law, B. H. Gwee, and J. Chang. Fast and memory-efficient invariant computation of ordinary petri nets. *IEE Proceedings: Computers and Digital Techniques*, 1(5):612–624, 2007.

[66] N. le Novère, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. Sauro, M. Schilstra, B. Shapiro, J. L. Snoep, and M. Hucka. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acid Research*, 1(34):D689–D691, Jan. 2006.

[67] A. Levchenko, J. Bruck, and P. W. Sternberg. Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. *PNAS*, 97(11):5818–5823, May 2000.

[68] R. J. Lipton. The reachability problem requires exponential space. Technical report 62, Yale University, 1976.

[69] F. Liu and M. Heiner. Colored petri nets to model and simulate biological systems. In *ACSD/Petri Nets Workshops 2010*, pages 71–85, 2010.

[70] J. Martinez and M. Silva. A simple and fast algorithm to obtain all invariants of a generalized petri net. In *Selected Papers from the First and the Second European Workshop on Application and Theory of Petri Nets*, pages 301–310, London, UK, UK, 1982. Springer-Verlag.

[71] W. Marwan, A. Sujatha, and C.Starostzik. Reconstructing the regulatory network controling commitment and sporulation in physarum polycephalum based on hierarchical petri net modeling and simulation. *Journal of Theoretical Biology*, 236:349–365, 2005.

[72] H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid petri net representation of gene regulatory network. In *Proceedings of the 5th Pacific Symposium on Biocomputing*, pages 338–349, 2000.

[73] H. Matsuno, Y. Tanaka, H. Aoshima, A. Doi, M. Matsui, and S. Miyano. Biopathways representation and simulation on hybrid functional petri net. *In Silico Biology*, 3:32, 2003.

[74] E. W. Mayr. An algorithm for the general petri net reachability problem. *SIAM Journal of Computing*, 13, 1984.

[75] M. Minoux and K. Barkaoui. Deadlocks and traps in petri nets as horn-satisfiability solutions and some related polynomially solvable problems. *Discrete Applied Mathematics*, 29:195–210, 1990.

[76] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of sat problems. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 459–465. AAAI press, 1992.

[77] I. Mura and A. Csiksz-Nasy. Stochastic petri net extension of a yeast cell cycle model. *Journal of Theoretical Biology*, 254::850–860, 2008.

[78] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–579, Apr. 1989.

[79] F. Nabli. Finding minimal siphons as a csp. In *CP'11: The Seventeenth International Conference on Principles and Practice of Constraint Programming, Doctoral Program*, pages 67–72, Sept. 2011.

[80] F. Nabli, T. Martinez, F. Fages, and S. Soliman. Finding siphons in petri-nets: Complexity and algorithms (in preparation). *Constraints*, 2013.

[81] O. Oanea, H. Wimmel, and K. Wolf. New algorithms for deciding the siphon-trap property. In *PETRI NETS'10 Proceedings of the 31st international conference on Applications and Theory of Petri Nets*, pages 267–286. Springer-Verlag, 2010.

[82] K. Oda, Y. Matsuoka, A. Funahashi, and H. Kitano. A comprehensive pathway map of epidermal growth factor receptor signaling. *Molecular Systems Biology*, 1, May 2005.

[83] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, New Jersey, 1981.

[84] A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[85] V. N. Reddy, M. L. Mavrovouniotis, and M. N. Liebman. Petri net representations in metabolic pathways. In L. Hunter, D. B. Searls, and J. W. Shavlik, editors, *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 328–336. AAAI Press, 1993.

[86] N. Robertson and P. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[87] S. Roch and P. P. H. Starke. *INA Integrated Net Analyzer Version 2.2 Manual*. Humboldt-Universitat zu Berlin, Institut fur Informatik, 2000.

[88] A. Sackmann, D. Formanowicz, P. Formanowicz, I. Koch, and J. Blazewicz. An analysis of the petri net based model of the human body iron homeostasis process. *Computational Biology and Chemistry*, 31(1):1–10, 2007.

[89] A. Sackmann, M. Heiner, and I. Koch. Application of petri net based analysis techniques to signal transduction pathways. *BMC Bioinformatics*, 7(482), Nov. 2006.

[90] M. A. Savageau. Biochemical systems analysis i some mathematical properties of the rate law for the component enzymatic reactions. *Journal of Theoretical Biology*, 25(3):365–369, 1969.

[91] M. A. Savageau. Biochemical systems analysis ii the steady-state solutions for an n-pool system using a power-law approximation. *Journal of Theoretical Biology*, 25(3):370–379, 1969.

[92] B. Schoeberl, C. Eichler-Jonsson, E. Gilles, and G. Muller. Computational modeling of the dynamics of the map kinase cascade activated by surface and internalized egf receptors. *Nature Biotechnology*, 20(4):370–375, 2002.

[93] S. Soliman. Finding minimal P/T-invariants as a CSP. In *Proceedings of the fourth Workshop on Constraint Based Methods for Bioinformatics WCB'08, associated to CPAIOR'08*, May 2008.

[94] S. Soliman. Modelling biochemical reaction networks with biocham extracting qualitative and quantitative information from the structure. In *Proceedings of the 6th Vienna Conference on Mathematical Modelling MATHMOD'09*, volume 35, pages 2304–2312. ARGESIM, Feb. 2009.

[95] S. Soliman. Invariants and other structural properties of biochemical models as a constraint satisfaction problem. *Algorithms for Molecular Biology*, 7(15), May 2012.

[96] N. Soranzo and C. Altafini. ERNEST: a toolbox for chemical reaction network theory. *Bioinformatics*, 25(21):2853–2854, 2009.

[97] R. Srivastava, M. Peterson, and W. Bentley. Stochastic kinetic analysis of the escherichia coli stress circuit using $\sigma^{32}$-targeted antisense. *Biotechnology and Bioengineering*, 75:120–129, 2001.

[98] L. J. Steggles, R. Banks, and A. Wipat. Modelling and analysing genetic networks: From boolean networks to petri nets. In *CMSB06, LNCS 4210*, pages 127–141. Springer, 2006.

[99] L. Stryer. *Biochemistry*. Freeman, New York, 1995.

[100] Z. Szallasi, J. Stelling, and V. Periwal, editors. *System Modeling in Cellular Biology: From Concepts to Nuts and Bolts*. MIT Press, 2006.

[101] S. Tanimoto, M. Yamauchi, and T. Watanabe. Finding minimal siphons in general petri nets. *IEICE Trans. on Fundamentals in Electronics, Communications and Computer Science*, pages 1817–1824, 1996.

[102] E. O. Voit. *Computational Analysis of Biochemical Systems. A Practical Guide for Biochemists and Molecular Biologists*. Cambridge University Press, Cambridge, U.K., 2000.

[103] A. von Kamp and S. Schuster. Metatool 5.0: fast and flexible elementary modes analysis. *Bioinformatics*, 22(15):1930–1931, 2006.

[104] M. Wallace. Practical applications of constraint programming. *Constraints*, 1(1/2):139–168, 1996.

[105] M. Yamauchi, M. Wakuda, S. Taoka, and T. Watanabe. A fast and space-saving algorithm for computing invariants of petri nets. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 1999. IEEE SMC '99*, volume 1, pages 866–871, 1999.

[106] M. Yamauchi and T. Watanabe. Time complexity analysis of the minimal siphon extraction problem of petri nets. *EICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, pages 2558–2565, 1999.

[107] I. Zevedei-Oancea and S. Schuster. Topological analysis of metabolic networks based on petri net theory. *In Silico Biology*, 3(29), 2003.