Hybrid Petri Nets State Space Representation Using Coverability Graphs and Unfoldings

FIT BUT Technical Report Series

Petr Novosad and Milan Češka

Technical Report No. FIT-TR-2015-01
Faculty of Information Technology, Brno University of Technology

Last modified: June 15, 2015

Abstract

This technical report deals with continuous and hybrid Petri nets state space representation using coverability graphs and unfoldings. The coverability graph, resp. unfolding are methods for Petri nets analysis that can represent an infinite state space of an unbounded Petri net with finite graph, resp. net. These techniques can cope well with the so-called state space explosion problem. Formalizations of the representations are presented together with algorithms for their computing and typical examples.

Keywords:

Continuous Petri nets, Hybrid Petri nets, Coverability graphs, Unfoldings.

Acknowledgement

This work has been supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

Contents

Chapter 1

Introduction

1.1 Motivation and Goals

Hybrid Petri nets [1] have many modern commercial applications. One of the best examples is simulation of biological pathways in human cells [2, 3]. Although an extended hybrid Petri net with time is used in these applications, an autonomous hybrid Petri net is also worth studying. Such autonomous Petri nets describe the system in a qualitative manner and we can run useful analysis and verification on them.

Analysis options of autonomous hybrid Petri nets have been discussed with Hassane Alla the co-author of hybrid Petri nets during several visits to University in Grenoble. Because of the continuous change of the marking, even bounded hybrid Petri nets have infinite state space and thus similar set of problems with reachability arises as for unbounded discrete Petri nets [4]. There exists several techniques to represent infinite state space with finite graphs or nets for discrete Petri nets. Notably coverability graphs [5] and unfoldings [6]. These methods use abstraction and aggregation of markings. Each of them bring its own way of view on the state space. Both approaches can make analysis of the hybrid Petri nets easier.

The overall goals are to adapt the coverability graphs and unfoldings from the discrete Petri nets to the continuous and hybrid Petri nets, to define the notation of the coverability graph for hybrid Petri nets, develop an algorithm for their computation and create some examples, and to define the notation of the unfoldings of hybrid Petri nets, develop an algorithm for their computation and again create some examples.

Finally to compare the two approaches with listing advantages and drawbacks on typical examples, case analysis and verification options.

1.2 Related Work

Authors of hybrid Petri nets present several examples of coverability graphs for continuous and hybrid Petri nets in their book [1] and paper [7]. But the authors present it without any formalization or any formal algorithm only as an illustration that such representation

is possible.

More detailed information about related work regarding coverability graphs and unfolding of discrete Petri nets is described in the next chapter 3.

1.3 Structure of the Report

The report is organized into 5 chapters as follows:

1. *Introduction*: Describes the motivation behind our efforts together with our goals.
2. *Background and State-of-the-Art*: Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art.
3. *Overview of Our Approach*: Presents our work on formalization, algorithms and examples of coverability graphs and unfoldings for continuous and hybrid Petri nets.
4. *Comparison of Results*: Compares the two approaches, lists advantages and drawbacks on examples.
5. *Conclusions*: Summarizes the results of our research, suggests topics for further research, and concludes the report.

Chapter 2

Background and State-of-the-Art

2.1 Discrete Petri nets

Petri nets are a mathematical and graphical tool for modeling concurrent, parallel and/or distributed systems. This report assumes that the reader is familiar with the basic theory of the Petri nets [8, 9].

Discrete Petri net [1] is defined as a 5-tuple $R_D = (P, T, Pre, Post, M_0)$, where P is a finite set of places and T is a finite set of transitions. $P \neq \emptyset$, $T \neq \emptyset$ and $P \cap T = \emptyset$. $Pre\colon P \times T \to \mathbb{N}$ is the input incidence matrix. $Post\colon P \times T \to \mathbb{N}$ is the output incidence matrix. $M_0\colon P \to \mathbb{N}$ is the initial marking. Let $p \in P, t \in T : Pre(p,t)$ is the weight of the arc $p \to t$; $Post(p,t)$ is the weight of the arc $t \to p$. If the arc does not exist, the weight is 0. In a graphical representation of the discrete Petri net places are represented by circles and transitions are represented by rectangles (see Fig. 2.1).

The *discrete marking* $m \in (\mathbb{N})^{|P|}$ is a vector of natural numbers. A transition $t \in T$ is *enabled* in a marking m, iff $\forall p \in {}^\bullet t : m(p) > Pre(p,t)$. Firing the transition t with a quantity $\alpha \in \mathbb{N}$ is denoted as $m \xrightarrow{\alpha t} m'$. $[t]^\alpha$ represents $\alpha \in \mathbb{N}$ firings of the transition t at one go. The new marking $m' = m + \alpha.C(P,t)$, where $C = Post - Pre$ is a token-flow matrix. The marking m' is *reachable* from the marking m.

A Petri net is *bounded* if the number of tokens in all places and in all reachable markings is less than some upper bound. The Petri net is *persistent* when enabled transitions can only be disabled by its own firing. For more analysis options see [8].

2.2 Continuous Petri Nets

The concept of the continuous and hybrid Petri nets has been presented by David and Alla in 1987 [10, 11, 1, 12]. It is a fluidification of the discrete Petri net. Some places can hold a real valued marking.

Continuous Petri net [1] is defined as a 5-tuple $R_C = (P, T, Pre, Post, M_0)$, where P is a finite set of places and T is a finite set of transitions. $P \neq \emptyset$, $T \neq \emptyset$ and $P \cap T = \emptyset$. $Pre : P \times T \to \mathbb{Q}^+$ is the input incidence matrix. $Post : P \times T \to \mathbb{Q}^+$ is the output

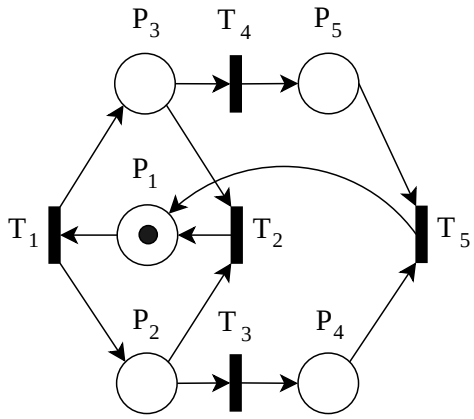

Figure 2.1: The bounded discrete Petri net.

incidence matrix. $M_0 : P \to \mathbb{R}^+$ is the initial marking[1]. Let $p \in P, t \in T : Pre(p,t)$ is the weight of the arc $p \to t$; $Post(p,t)$ is the weight of the arc $t \to p$. If the arc does not exist, the weight is 0. In a graphical representation of the continuous Petri net places are represented by double circles and transitions are represented by empty rectangles (see Fig. 2.2).

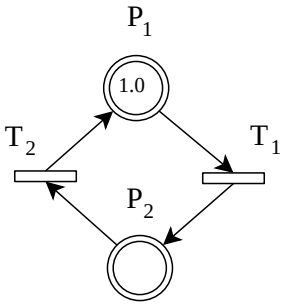

Figure 2.2: The bounded continuous Petri net.

The *continuous marking* $m \in (\mathbb{R}^+)^{|P|}$ is a vector of non-negative real numbers. A transition $t \in T$ is *enabled* in a marking m, iff $\forall p \in {}^\bullet t : m(p) > 0$. Enabling of the transition does not depend on the arc weight, it is sufficient that every input place has a non-zero marking. The *enabling degree* q of the transition t for the marking m is the maximal amount that the transition can fire in one go, i.e. $q(t,m) = \min_{p \in {}^\bullet t} (m(p)/Pre(p,t))$. Firing the transition t with a quantity $\alpha < q(t,m), \alpha \in \mathbb{R}^+$ is denoted as $m \xrightarrow{\alpha t} m'$. $[t]^\alpha$ represents $\alpha \in \mathbb{R}^+$ firings of the transition t at one go. The new marking $m' = m + \alpha.C(P,t)$, where $C = Post - Pre$ is a token-flow matrix. The marking m' is *reachable* from the marking m.

Let m be a marking. The set P of places may be divided into two subsets: $P^+(m)$ the set of places $p \in P$ such that $m(p) > 0$, and the set of places p such that $m(p) = 0$.

[1]Notation $\mathbb{Q}^+$ corresponds to the non-negative rational numbers and notation $\mathbb{R}^+$ corresponds to the non-negative real numbers (both including zero).

A *continuous macro-marking* is the union of all markings m with the same set $P^+(m)$ of marked places. Since each continuous macro-marking is based on the Boolean state of every place (marked or not marked), the number of continuous macro-markings is less than or equal to 2^n, where n is the number of places.

2.3 Hybrid Petri Nets

Hybrid Petri net [1] is a 6-tuple $R_H = (P, T, Pre, Post, M_0, h)$, where P is a finite set of discrete and continuous places, T is a finite set of discrete and continuous transitions. $P \neq \emptyset$, $T \neq \emptyset$ and $P \cap T = \emptyset$. $Pre : P \times T \to \mathbb{Q}^+$ or $\mathbb{N}$ is the input incidence matrix. $Post : P \times T \to \mathbb{Q}^+$ or $\mathbb{N}$ is the output incidence matrix. Let $p \in P, t \in T : Pre(p,t)$ is the weight of the arc $p \to t$; $Post(p,t)$ is the weight of the arc $t \to p$. If the arc does not exist, the weight is 0. A graphical representation of the hybrid Petri net is shown in Fig. 3.5. $M_0 : P \to \mathbb{R}^+$ or $\mathbb{N}$ is the *initial marking*. A function $h : P \cup T \to \{D, C\}$ is called a *hybrid function*, that indicates for every node whether it is a discrete node (sets P_D and T_D) or a continuous one (sets P_C and T_C). In the definitions of $Pre, Post$ and m_0, the set $\mathbb{N}$ corresponds to the case where $p \in P_D$ and the set $\mathbb{Q}^+$ to the case where $p \in P_C$. For the discrete places $p \in P_D$ and the continuous transitions $t \in T_C$ must hold $Pre(p,t) = Post(p,t)$.

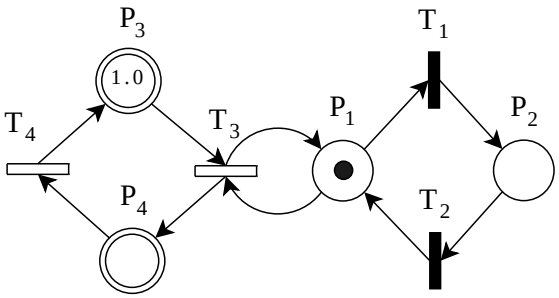

Figure 2.3: The bounded hybrid Petri net.

The *hybrid marking* for the hybrid Petri net is a couple $m = (m_C, m_D)$, where m_C denotes the continuous macro-marking of the continuous places and m_D denotes the marking of the discrete places. The discrete transition $t \in T_D$ is *enabled* in a marking m, iff $\forall p \in {}^\bullet t : m(p) \geq Pre(p,t)$. The *enabling degree* q of the discrete transition t for the marking m is integer $q(t,m) = \min_{p \in {}^\bullet t} (m(p)/Pre(p,t))$. For continuous places $p \in {}^\bullet t \wedge p \in P_C$ the edge $p \to t$ is a treshold for marking in the place p for enabling the discrete transition t. A continuous transition $t \in T_C$ is *enabled* in a marking m, iff $\forall p \in {}^\bullet t \wedge p \in P_D : m(p) \geq Pre(p,t)$ and $\forall p \in {}^\bullet t \wedge p \in P_C : m(p) > 0$. The *enabling degree* q of the continuous transition t for the marking m is $q(t,m) = \min_{p \in {}^\bullet t} (m(p)/Pre(p,t))$.

The hybrid Petri net in Fig. 2.3 shows a typical example of interaction between discrete and continuous parts of a hybrid Petri net. The discrete part serves as a 'switch' for the continuous part. The place P_1 enables or disables the transition T_3. There are however

other ways how the continuous and discrete parts can interact with each other in the hybrid Petri net. All the possible combinations are described in [1].

2.4 Coverability Graphs

For the discrete Petri nets there exist an algorithm developed by Karp and Miller [13] for computing a *reachability root tree*. The *reachability graph* is obtained by merging all the vertices of the reachability root tree corresponding to the same marking. The *coverability graph* gathers nodes which correspond to the same marking [5] using a symbol ω that represents arbitrarily many tokens. The state space of the unbounded Petri net is there presented as a finite graph $G_d = (N, E)$. Nodes $N \subseteq (\mathbb{N} \cup \omega)^{|P|}$ in the coverability graph represent states of a system and edges $E \subseteq N \times T \times N$ represent transition firings.

There exist algorithms for checking qualitative properties such as safeness, boundness, conservativeness, coverability and reachability for the bounded Petri nets from the reachability graphs. However one cannot check properties such as deadlock, liveness and reachability for the unbounded Petri nets from the coverability graphs because of the aggregation of markings and thus the loss of information.

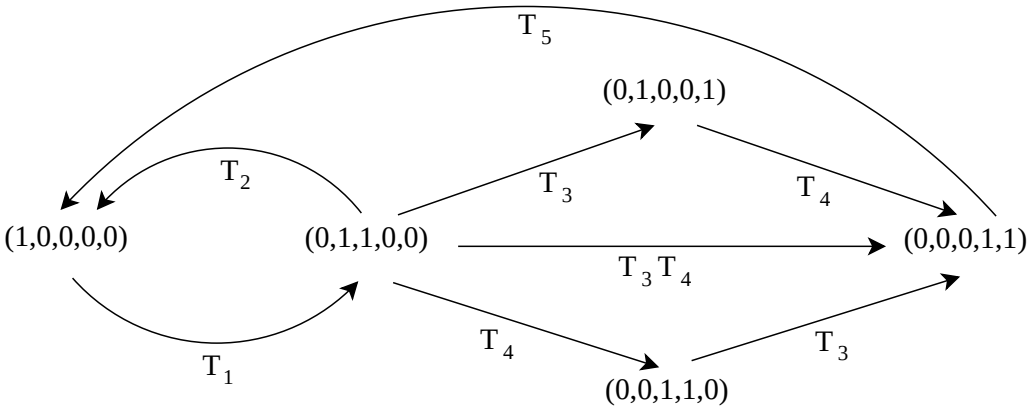

Figure 2.4: The coverability graph of the discrete Petri net in Fig. 2.1.

The Fig. 2.4 shows an example of the reachability graph constructed for the bounded discrete Petri net in Fig. 2.1. All states of the system and all possible transition firings are clearly visible there.

2.5 Unfoldings

The unfolding [14, 15, 16, 17, 18] is a useful partial-order method for analysis and verification of the Petri net properties. The state space of the Petri net is represented by an acyclic net with a simpler structure than the Petri net.

A *net* is a triple $N = (P, T, F)$, where P is a finite set of places and T is a finite set of transitions. $P \neq \emptyset$, $T \neq \emptyset$ and $P \cap T = \emptyset$. $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation.

An *occurrence net* is a net $O = (B, E, G)$, where B is a set of occurrence of places, E is a set of occurrence of transitions. O is acyclic and G is the acyclic flow relation, i.e. for every $x, y \in B \cup E : xG^+y \Rightarrow \neg yG^+x$, where G^+ is a transitive closure of G. Let us denote $x < y$, iff xG^+y, and $x \leq y$, iff $x < y$ or $x = y$. The relation $<$, resp. $\leq$ is a partial order relation. Nodes $x, y \in (P \cup T)$ are in a *conflict* relation, denoted by $x\#y$, iff $\exists t_1, t_2 \in T : t_1 \neq t_2 \wedge {}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset \wedge t_1 \leq x \wedge t_2 \leq y$. Nodes $x, y \in (P \cup T)$ are in a *concurrency* relation, denoted by $x \; co \; y$, if neither $x < y$ nor $y < x$ nor $x\#y$. For every $b \in B : |{}^\bullet b| \leq 1$. For every $x \in (B \cup E) : \neg(x\#x)$, i.e. no element is in conflict with itself. The set of elements $\{y \in (B \cup E) | y < x\}$ is finite, i.e. O is finitely preceded. $Min(O)$ denotes the set of minimal elements of $B \cup E$ with respect to the relation $\leq$, i.e. the elements with an empty preset.

A *homomorfism* from the occurrence net O to the discrete Petri net $R_D = (P, T, Pre, Post, M_0)$ is a mapping $p : B \cup E \to P \cup T$ such that $p(B) \subseteq P$ and $p(E) \subseteq T$, i.e. preserves the nature of nodes. For every $e \in E : p({}^\bullet e) =^\bullet p(e) \wedge p(e^\bullet) = p(e)^\bullet$, i.e. p preserves the environment of transitions. The restriction of p to $Min(O)$ is a bijection between $Min(O)$ and M_0.

A *branching process* of the discrete Petri net R_D is a 4-tuple $\pi_H = (B, E, G, p) = (O, p)$, where O is the labeled occurrence net and $p(x) = y$ denotes labeling element x as element y.

A branching process ${\pi'}_D = (O', p')$ is a *prefix* of π_D, denoted by ${\pi'}_D \sqsubseteq \pi_D$, if $O' = (B', E', G')$ is a subnet of O satisfying $Min(O)$ belongs to O'; if $e \in E'$ and $(b, e) \in G$ or $(e, b) \in G$ then $b \in B'$; if $b \in B'$ and $(e, b) \in G$ then $e \in E'$; p' is the restriction of p to $B' \cup E'$. For every R_D there exists a unique (up to isomorphism) maximal (w.r.t. $\sqsubseteq$) branching process that is called *unfolding*.

A *configuration* of the occurrence net O is a set of the transitions occurrences $C \subseteq E$ such that for all $e_1, e_2 \in C : \neg(e_1\#e_2)$, i.e. C is conflict-free. For every $e_1 \in C : e_2 \leq e_1 \Rightarrow e_2 \in C$, i.e. C is causally closed. A *local configuration* $[e]$ for the transition occurrence $e \in E$ is a set $[e] = \{e' \in E | e' \leq e\}$.

A set of places occurrences $D \subseteq B$ is called a *co-set*, iff for all distinct $d_1, d_2 \in D : d_1 \; co \; d_2$. A *cut* is the maximal (w.r.t. set inclusion) co-set. For every $d_1, d_2 \in D$, if $p(d_1) = p(d_2)$ then $d_1 = d_2$. Let C be the finite configuration of the branching process π_D. Then $Cut(C) = (Min(O) \cup C^\bullet) \setminus {}^\bullet C$ is a cut. A set $Mark(C) = p(Cut(C))$ is the reachable marking of the discrete Petri net R_D.

An *adequate order* $\lhd$ is a strict well-founded partial order on the local configurations such that for two transitions occurrences $e_1, e_2 \in E : [e_1] \subset [e_2] \Rightarrow [e_1] \lhd [e_2]$. The transition occurrence $e_1 \in E$ is a *cut-off* transition induced by $\lhd$, iff there is a corresponding transition $e_2 \in E$ with $Mark([e_1]) = Mark([e_2])$ and $[e_2] \lhd [e_1]$. The order $\lhd$ is a refined partial order from [14].

The branching process is *complete*, iff for every reachable marking $M \in [M_0 >$ of the discrete Petri net R_D there is the configuration C of π_H such that $M = Mark(C)$ and for every transition $t \in T$ enabled in M there is the finite configuration C and the transition

occurrence $e \in C$ such that $M = Mark(C)$, $p(e) = t$ and $C \cup \{e\}$ is the configuration.

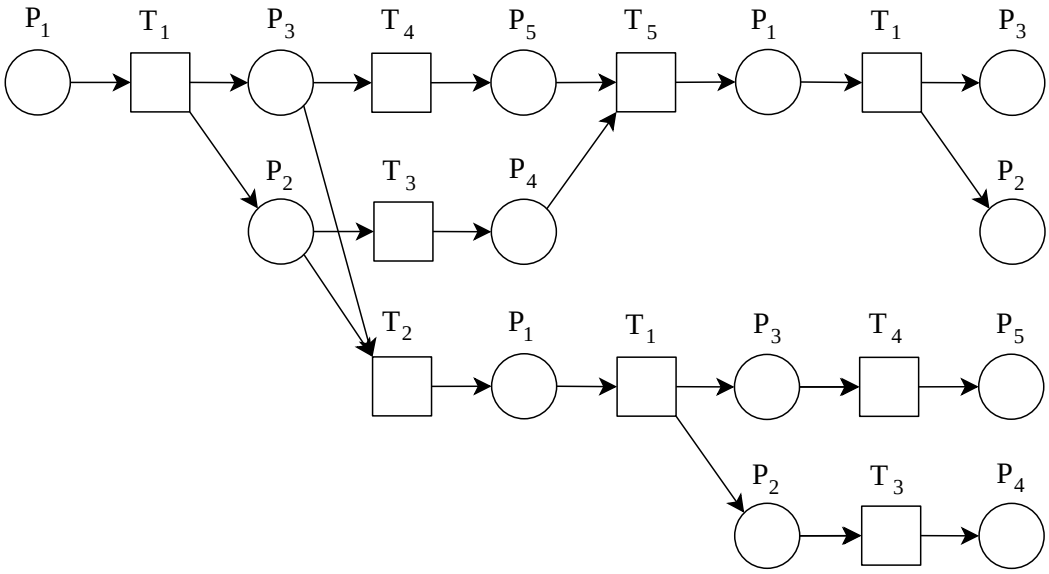

Figure 2.5: The branching process of the discrete Petri net in Fig. 2.1.

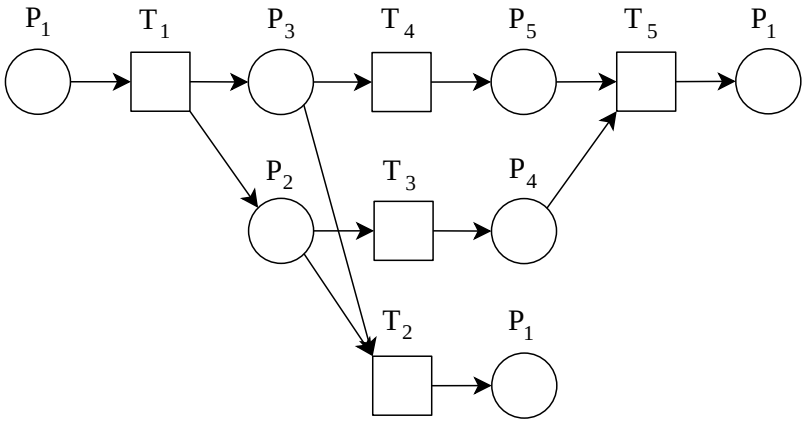

Figure 2.6: The coverability graph of the discrete Petri net in Fig. 2.6.

The Fig. 2.5 shows an example of the branching process constructed for the bounded discrete Petri net in Fig. 2.1. The branching process represents all reachable states of the Petri net and can be infinite if the Petri net has a cycle as in this example. However it can be truncated before it starts to repeat and the resulting unfolding is shown in Fig. 2.6.

Chapter 3

Overview of Our Approach

3.1 Coverability Graphs

Our approach adapts the coverability graphs from the discrete Petri nets to the continuous and hybrid Petri nets. Continuous conditions in the coverability graphs of the hybrid Petri net can have associated a symbol representing the macro-marking thus some nonzero real marking. Discrete conditions in the coverability graphs of the hybrid Petri net can have associated a symbol ω representing that the corresponding place is unbounded.

We have formalized following notations of the coverability graphs of the continuous and hybrid Petri nets and developed following algorithms [19, 20] for their computation.

3.1.1 Coverability Graphs of Bounded Continuous Petri Nets

The *coverability graph* G_{cb} of the bounded continuous Petri net $R_{CB} = (P, T, Pre, Post, M_0)$ is a pair $G_{CB} = (N, E)$. The set $N \subseteq (\mathbb{R}^+ \cup \{m_1, \ldots, m_{|P|}\})^{|P|}$ is the set of states, where $m_i,\ i = 1 \ldots |P|$ is a substitute symbol that represents non-zero marking in the continuous place p_i. The elements of N are the macro-markings. The set $E \subseteq N \times T \times ((\mathbb{R}^+ \setminus \{0\}) \cup \{m_1, \ldots, m_{|P|}\}) \times N$ is the set of edges labeled with a fired transition and its degree. The degree can be substituted with the symbol m_i. Firing a transition without specifying the firing quantity means that it was fired with less quantity than the maximal enabling degree.

The algorithm 1 constructs the coverability graph for the bounded continuous Petri net. A function `AddNewNode()` adds a new node for the given marking and flags it as *unprocessed*. A function `GetEnabledTransitions()` returns a set of enabled transitions for the macro-marking represented by the given node. A function `GetEnablingDegrees()` returns a set of enabling degrees that are valid for the given transition and node. It returns values of a maximal enabling degree and a half of the maximal enabling degree. A function `GetNode()` returns an existing node for the given marking. A function `FireTransition()` returns a new macro-marking after firing the given transition with the given degree in the macro-marking represented by the given node. The new macro-marking m is created as follows. If a marking in the place p_i is a boundary value (including zero) then the marking

$m(p_i)$ is set to this value. Else the marking $m(p_i)$ is set to the substitute symbol m_i. The substitute symbol m_i for the place p_i need not to propagate to the following marking. Basically, new nodes are created when a marking of a place becomes zero, or when an unmarked place becomes marked.

Algorithm 1: The coverability graph for the bounded continuous Petri nets.

Input: The bounded continuous Petri net $R_{CB} = (P, T, Pre, Post, M_0)$
Output: The coverability graph $G_{CB} = (N, E)$
Method:
begin
  `AddNewNode(`M_0`);`
  while *exists a node* $n \in N$ *such that* n *is unprocessed* **do**
    Flags the node n as processed;
    $F =$`GetEnabledTransitions(`n`);`
    for *each transition* $t \in F$ **do**
      $Q =$`GetEnablingDegrees(`n, t`);`
      for *each degree* $q \in Q$ **do**
        $m' =$`FireTransition(`n, t, q`);`
        if *a node with* m' *does not exist in* N **then**
          `AddNewNode(`m'`);`
        end
        $n' =$`GetNode(`m'`);`
        if *an edge* (n, t, q, n') *does not exist in* E **then**
          `AddNewEdge(`n, t, q, n'`);`
        end
      end
    end
  end
end

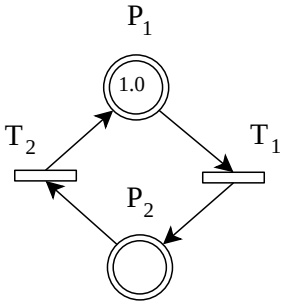

Figure 3.1: The bounded continuous Petri net.

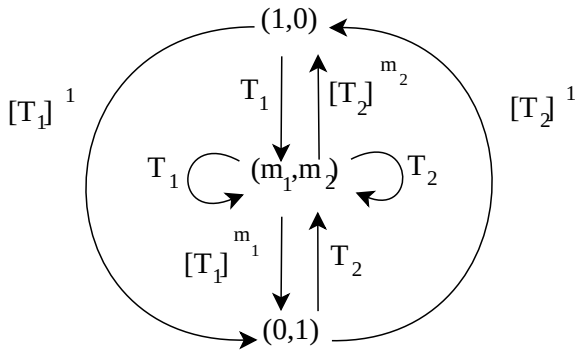

Figure 3.2: The coverability graph of the continuous Petri net in Fig.3.1.

An example of the coverability graph created by the algorithm 1 for the bounded continuous Petri net in Figure 3.1 is in Figure 3.2. The net has two places, therefore it can has a maximum of 4 macro-markings. Reachable macro-markings are $(1, 0), (m_1, m_2)$ and $(0, 1)$.

3.1.2 Coverability Graphs of Unbounded Continuous Petri Nets

The coverability graph of the unbounded continuous Petri net is a pair $G_{CU} = (N, E)$, where $N \subseteq (\mathbb{R}^+ \cup \{\omega\} \cup \{m_1, \ldots, m_{|P|}\})^{|P|}$. The set E is the same as for G_{CB}. The symbol ω represents an arbitrarily large number in a place and the marking in a such place is unbounded.

Algorithm 2: The coverability graphs for the unbounded continuous Petri nets.

Input: The unbounded continuous Petri net $R_{CU} = (P, T, Pre, Post, M_0)$
Output: The coverability graph $G_{CU} = (N, E)$
Method:
This algorithm is similar to the algorithm 1. The main difference is in the function `FireTransition()`, that can return the macro-markings with the symbol ω. The macro-marking m'' depends on the previous macro-marking m' as follows. If the macro-marking m'' covers the macro-marking m' ($m'' > m'$) then for places where $m''(p_i) > m'(p_i)$ set $m''(p_i) = \omega$. The symbol ω in the place p_i must propagate to the succeeding macro-markings in the place p_i.

An example of the coverability graph created by the algorithm 2 for the unbounded continuous Petri net in Figure 3.3 is in Figure 3.4. The system in Figure 3.3 has a self-loop and the place P_2 is unbounded. This is projected to the macro-markings $(1, \omega, 0), (0, \omega, 1)$ and (m_1, ω, m_3) in Figure 3.4.

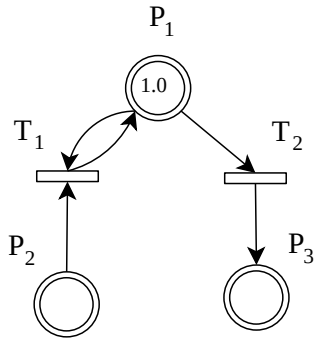

Figure 3.3: The unbounded continuous Petri net.

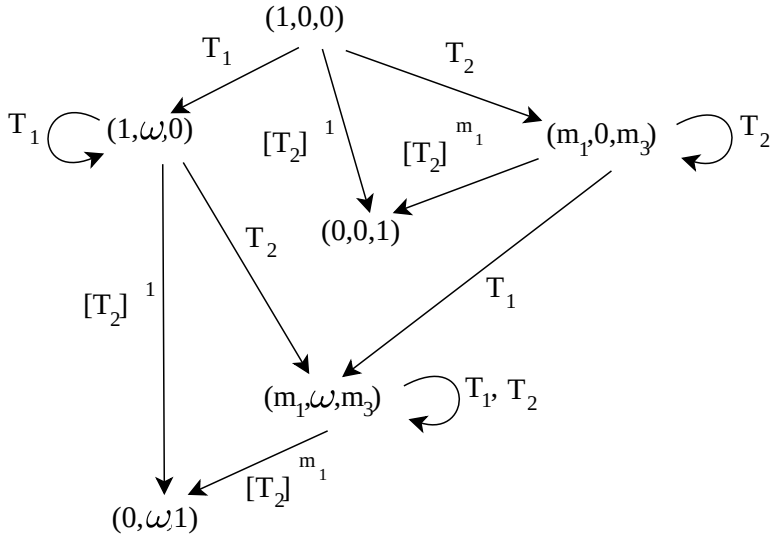

Figure 3.4: The coverability graph of the unbounded continuous Petri net in Fig.3.3.

3.1.3 Coverability Graphs of Bounded Hybrid Petri Nets

The coverability graph of the bounded hybrid Petri net $R_{HB} = (P, T, Pre, Post, M_0, h)$ is a pair $G_{HB} = (N, E)$. The set $N \subseteq (\mathbb{R}^+ \cup \{m_1, \ldots, m_{|P_C|}\})^{|P_C|} \times \mathbb{N}^{|P_D|}$ is the set of states, where $m_i,\ i = 1 \ldots |P_C|$ is a substitute symbol that represents non-zero marking in the continuous place p_i. The elements of N are hybrid markings. The set $E \subseteq N \times T \times ((\mathbb{Q}^+ \setminus \{0\}) \cup \{m_1, \ldots, m_{|P_C|}\}) \times N$ is the set of edges as in coverability graph G_{CB} of the continuous hybrid Petri net.

An example of the coverability graph created by the algorithm 3 for the bounded hybrid Petri net in Figure 3.5 is in Figure 3.6. There is a simple connection between the continuous and the discrete part in the transition T_1. The coverability graph in Figure 3.6 shows the effect of marking and unmarking of the discrete place P_3 by the discrete transitions T_3 and T_4.

Algorithm 3: The coverability graph for the bounded hybrid Petri nets.

Input: The bounded hybrid Petri net $R_{HB} = (P, T, Pre, Post, M_0, h)$
Output: The coverability graph $G_{HB} = (N, E)$
Method:
This algorithm is similar to the algorithm 1. The main difference is that all functions work with hybrid markings. Moreover the following two functions need more detailed description. A function `GetEnablingDegrees()` returns a set of enabling degrees that are valid for the given transition and node. For the continuous transitions it returns values of maximal enabling degree and half of the maximal enabling degree. For the discrete transitions it returns the minimal enabling degree. A function `FireTransition()` returns a new hybrid marking after firing the given transition with the given degree in the hybrid marking represented by the given node. The new marking m_D is created according to the discrete Petri nets. The new continuous macro-marking m_C is created as follows. If a marking in the place p_i is a boundary value (including zero) then the marking $m_C(p_i)$ is set to this value. Else the marking $m_C(p_i)$ is set to the substitute symbol m_i. The substitute symbol m_i for the place p_i need not to propagate to the following marking.
Basically, new nodes are created when: a marking of a continuous place becomes zero, or an unmarked continuous place becomes marked, or a discrete transition is fired, or a change of a marking of a continuous place causes the change of an enabling degree of a discrete transition. Since the number of the continuous macro-markings is finite, the algorithm terminates when all the discrete markings and all the continuous macro-markings are explored.

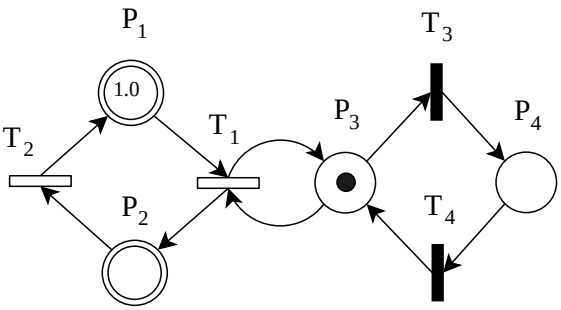

Figure 3.5: The bounded hybrid Petri net.

3.1.4 Coverability Graphs of Unbounded Hybrid Petri Nets

The coverability graph of the unbounded hybrid Petri net is a pair $G_{HU} = (N, E)$, where the set $N \subseteq (\mathbb{R}^+ \cup \{\omega\} \cup \{m_1, \ldots, m_{|P_C|}\})^{|P_C|} \times (\mathbb{N} \cup \omega)^{|P_D|}$. The set E is the same as for G_{HB}. The symbol ω represents an arbitrarily large number in a place and the marking in a such place is unbounded.

An example of the coverability graph created by the algorithm 4 for the unbounded

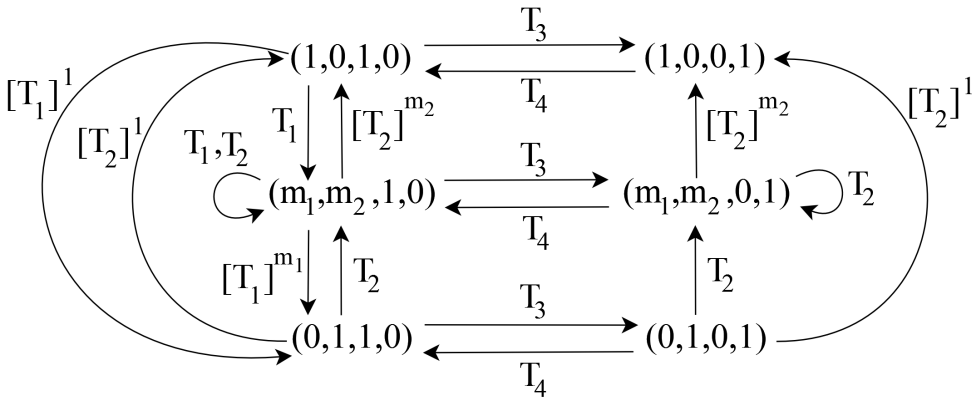

Figure 3.6: The coverability graph of the bounded hybrid Petri net in Fig.3.5.

Algorithm 4: The coverability graph for the unbounded hybrid Petri nets.

Input: The unbounded hybrid Petri net $R_{HU} = (P, T, Pre, Post, M_0, h)$
Output: The coverability graph $G_{HU} = (N, E)$
Method:
This algorithm is similar to the algorithm 3. The main difference is in the function `FireTransition()`, that can return continuous macro-markings and discrete macro-markings with the symbol ω. The new macro-marking m'' depends on the previous macro-marking m' as follows. If the macro-marking m'' covers the macro-marking m' ($m'' > m'$) then for places where $m''(p_i) > m'(p_i)$ set $m''(p_i) = \omega$. The symbol ω in the place p_i must propagate to the succeeding macro-markings in the place p_i.
Since the number of the continuous macro-markings and the number of discrete macro-markings is finite, the algorithm terminates when all the discrete macro-markings and all the continuous macro-markings are explored.

hybrid Petri net in Figure 3.7 is in Figure 3.8. The system in Figure 3.7 has self-loops in the continuous place P_1 and in the discrete place P_3. These places are unbounded. This is projected to the macro-markings in Figure 3.8 via symbol ω.

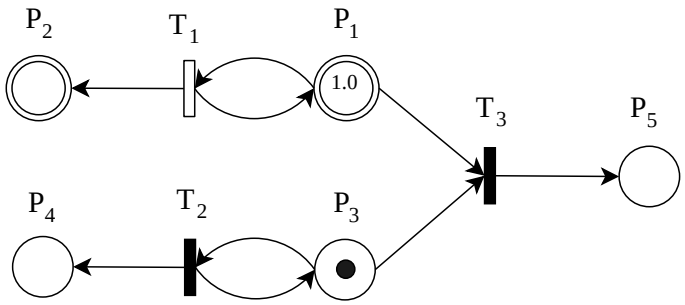

Figure 3.7: The unbounded hybrid Petri net.

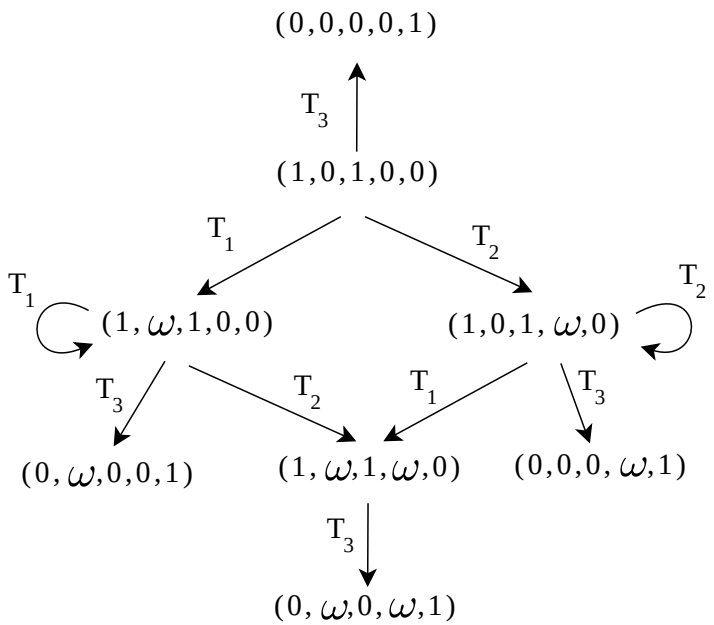

Figure 3.8: The coverability graph of the unbounded hybrid Petri net in Fig.3.7.

3.2 Unfoldings

Our approach combines the macro markings from the coverability graph for the continuous Petri nets [1, 7] with the idea of the coverability unfolding for the unbounded discrete Petri nets [6]. Continuous conditions in the unfolding can have associated a symbol representing the macro-marking thus some nonzero real marking. Discrete conditions in the unfolding can have associated a symbol ω representing that the corresponding place is unbounded.

We have formalized following notations [21] of the unfolding of the continuous and hybrid Petri nets and developed following algorithms [19] for their computation.

3.2.1 Unfoldings of Bounded Hybrid Petri Nets

The notation of the *net* $N = (P, T, F)$ and the *occurrence net* $O = (B, E, G)$ is described in chapter 2 section 2.5.

A *homomorfism* from the occurrence net O to the bounded hybrid Petri net $R_{HB} = (P, T, Pre, Post, M_0, h)$ is a mapping $p : B \cup E \to P \cup T$ such that $p(B) \subseteq P$ and $p(E) \subseteq T$, i.e. preserves the nature of nodes. For every $e \in E : p(^\bullet e) =^\bullet p(e) \wedge p(e^\bullet) = p(e)^\bullet$, i.e. p preserves the environment of transitions. The restriction of p to $Min(O)$ is a bijection between $Min(O)$ and M_0.

A *hybrid branching process* of the bounded hybrid Petri net R_{HB} is a 5-tuple $\pi_{HB} = (B, E, G, p, d) = (O, p, d)$, where O is the labelled occurrence net and $p(x) = y$ denotes labelling element x as element y. A mapping $d : E \to \{m_1, \ldots, m_{|P|}\} \cup \{0\}$ labels transitions occurrences with symbol m_i indicating maximal firing degree or with 0 indicating arbitrary lower degree (that will not be depicted). The type of the node determines its graphical representation. Every node $e \in E : p(e) \in T_C$ is represented by double rectangle and every node $b \in B : p(b) \in P_C$ is represented by double circle with the name of the corresponding marking.

A hybrid branching process $\pi'_{HB} = (O', p', d')$ is a *prefix* of π_{HB}, denoted by $\pi'_{HB} \sqsubseteq \pi_{HB}$, if $O' = (B', E', G')$ is a subnet of O satisfying $Min(O)$ belongs to O'; if $e \in E'$ and $(b, e) \in G$ or $(e, b) \in G$ then $b \in B'$; if $b \in B'$ and $(e, b) \in G$ then $e \in E'$; p' is the restriction of p to $B' \cup E'$. For every R_{HB} there exists a unique (up to isomorphism) maximal (w.r.t. $\sqsubseteq$) branching process that is called *unfolding*.

The notation of the *configuration* $C \subseteq E$, the *local configuration* $[e]$, the *co-set* $D \subseteq B$ and *cut* is described in chapter 2 section 2.5.

Let C be the finite configuration of the hybrid branching process π_{HB}. Then $Cut(C) = (Min(O) \cup C^\bullet) \setminus^\bullet C$ is a cut. A set $Mark(C) = p(Cut(C))$ is the reachable hybrid macro marking of the bounded hybrid Petri net R_{HB}.

The notation of the *adequate order* $\lhd$ is described in chapter 2 section 2.5. For the hybrid branching process π_{HB} and every $e_1, e_2 \in E : p(e_1) \in T_D \wedge p(e_2) \in T_C \Rightarrow [e_1] \lhd [e_2]$. For every $e_1, e_2 \in E : d(e_1) \neq 0 \wedge d(e_2) = 0 \Rightarrow [e_1] \lhd [e_2]$.

The hybrid branching process is *complete*, iff for every reachable hybrid macro marking $M \in [M_0 >$ of the bounded hybrid Petri net R_{HB} there is the configuration C of π_{HB} such that $M = Mark(C)$ and for every transition $t \in T$ enabled in M there is the finite

configuration C and the transition occurrence $e \in C$ such that $M = Mark(C)$, $p(e) = t$ and $C \cup \{e\}$ is the configuration.

Algorithm 5: The finite prefix of the unfolding for the bounded hybrid Petri net.

```
Input: The bounded hybrid Petri net R_HB = (P, T, Pre, Post, M_0, h)
Output: The finite prefix pref = (O, p, d) of the unfolding
begin
    InitializePrefix(pref);
    pe = PossibleExtensions(pref);
    cutoff = ∅;
    while pe ≠ ∅ do
        e = MinimalExtension(pe);
        if [e] ∩ cutoff = ∅ then
            Extend(pref, e);
            pe = PossibleExtensions(pref);
            if IsCutoff(e) then cutoff = cutoff ∪ {e};
        else
            pe = pe \ {e};
        end
    end
end
```

The algorithm 5 is a modified algorithm presented in [16]. It constructs the finite and complete prefix of the unfolding of the bounded hybrid Petri net. A function `InitializePrefix()` initializes the prefix *pref* with instances of the places from M_0. A function `PossibleExtensions()` finds the set of possible extensions of the branching process *pref* using possible transitions firings for the hybrid Petri net, including transitions firings with the maximal degree. The decision version of this function is NP-complete in the size of the prefix *pref*. A function `MinimalExtension()` chooses the transition occurrence with minimal local configuration with respect to the order $\lhd$ from the set of possible extensions. A function `Extend()` appends new instance of the transition occurrence and new instances of the output places of the transition. A function `IsCutoff()` determines whether the transition occurrence is a cut-off transition. Algorithm is finite because the number of macro markings in the bounded hybrid Petri net is finite and it transforms all transitions occurrences into cut-off transitions [15].

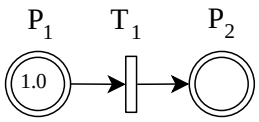

Figure 3.9: The bounded continuous Petri net.

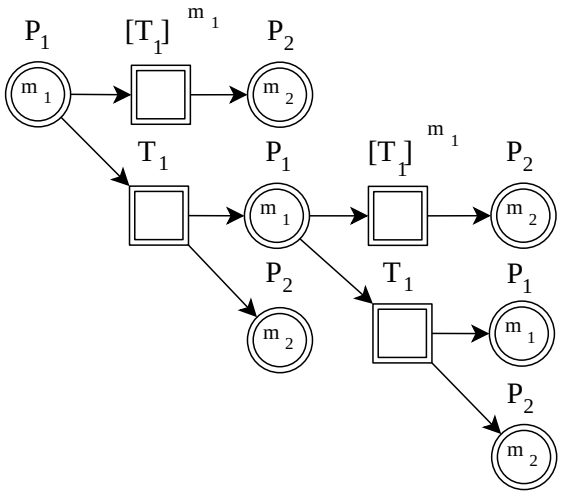

Figure 3.10: The finite prefix of the unfolding of the bounded continous Petri net from Fig. 3.9.

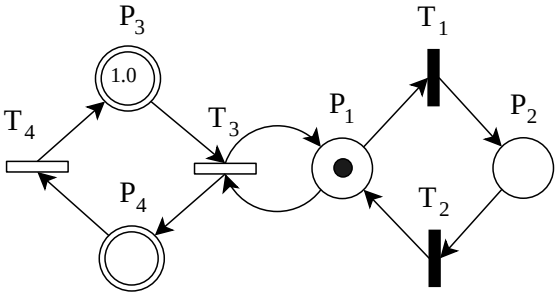

Figure 3.11: The bounded hybrid Petri net.

An example of the finite prefix of the unfolding created by the algorithm 5 for the bounded continuous Petri net in Fig. 3.9 is in Fig. 3.10. All reachable continuous macro markings are represented by cuts.

An example of the finite prefix of the unfolding created by the algorithm 5 for the bounded hybrid Petri net in Fig. 3.11 is in Fig. 3.12.

3.2.2 Unfoldings of Unbounded Hybrid Petri Nets

A hybrid branching process of the unbounded hybrid Petri net R_{HU} is a 6-tuple $\pi_{HU} = (B, E, G, p, d, w) = (O, p, d, w)$, where O, p, d are defined in the same way as in the previous section 3.2.1. A mapping $w : B \to \{\omega, 1\}$ labels discrete places occurrences with symbol ω indicating an unbounded discrete place or with 1 otherwise (that will not be depicted).

A hybrid branching process $\pi'_{HU} = (O', p', d', w')$ is a *prefix* of π_{HU}, denoted by $\pi'_{HU} \sqsubseteq \pi_{HU}$, if $O' = (B', E', G')$ is a subnet of O satisfying $Min(O)$ belongs to O'; if $e \in E'$ and $(b, e) \in G$ or $(e, b) \in G$ then $b \in B'$; if $b \in B'$ and $(e, b) \in G$ then $e \in E'$; p' is the restriction of p to $B' \cup E'$. For every R_{HU} there exists a unique (up to isomorphism)

maximal (w.r.t. $\sqsubseteq$) branching process that is called *unfolding*.

For the hybrid branching process π_{HU} and every $e_1, e_2 \in E : p(e_1) \in T_D \wedge p(e_2) \in T_C \Rightarrow [e_1] \lhd [e_2]$. For every $e_1, e_2 \in E : d(e_1) \neq 0 \wedge d(e_2) = 0 \Rightarrow [e_1] \lhd [e_2]$.

The hybrid branching process is complete, iff for every reachable hybrid macro marking $M \in [M_0 >$ of the unbounded hybrid Petri net R_{HU} there is the configuration C of π_{HU} such that $M = Mark(C)$ and for every transition $t \in T$ enabled in M there is the finite configuration C and the transition occurrence $e \in C$ such that $M = Mark(C)$, $p(e) = t$ and $C \cup \{e\}$ is the configuration.

Algorithm 6: The finite prefix of the unfolding for the unbounded hybrid Petri net.

Input: The unbounded hybrid Petri net $R_{HU} = (P, T, Pre, Post, M_0, h)$
Output: The finite prefix $pref = (O, p, d, w)$ of the unfolding
Method:
This algorithm is similar to the algorithm 5. The main difference is in the function `Extend()` which detects an unbounded discrete place by comparing the new and the previous discrete state and thus can return a discrete macro-markings with the symbol ω. The label of the unbounded discrete place is propagated further once denoted.
The algorithm is finite because the number of continuous, resp. discrete macro markings in the continuous, resp. discrete part of the hybrid Petri net is finite and it transforms all transitions occurrences into cut-off transitions [15].

An example of the finite prefix of the unfolding created by the algorithm 6 for the unbounded continuous Petri net in Fig. 3.13 is in Fig. 3.15. All reachable continuous macro markings are represented by cuts.

The image in Fig. 3.14 shows very simple, yet typical example from the application domain of the hybrid Petri nets, where the discrete part enables or disables the continuous transitions. An example of the complete and finite prefix of the unfolding created by the algorithm 6 for the unbounded hybrid Petri net in Fig. 3.14 is in Fig. 3.16. The image shows only the most interesting part of the whole prefix because of size limitations. It can be seen how the unbounded discrete place is detected and propagated further.

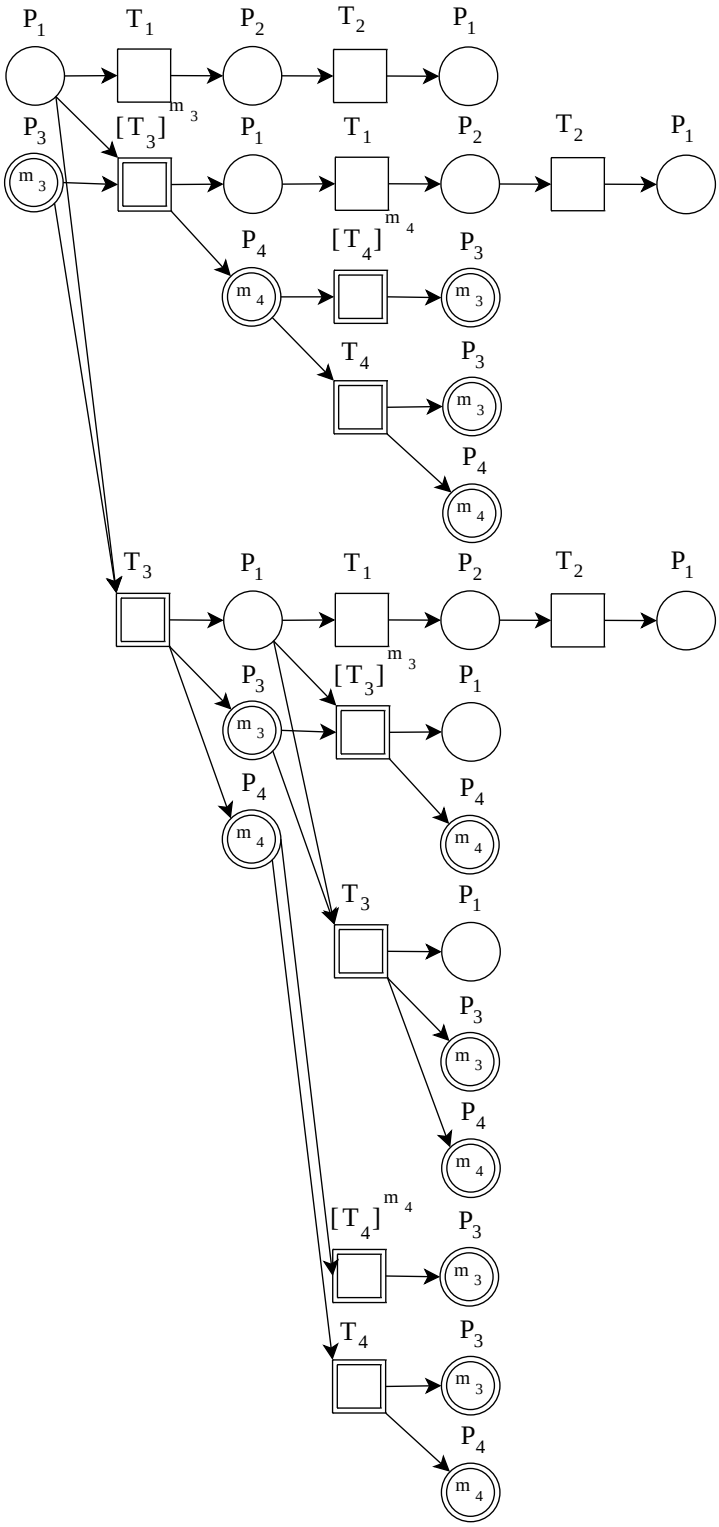

Figure 3.12: The finite prefix of the unfolding of the bounded hybrid Petri net from Fig. 3.11.

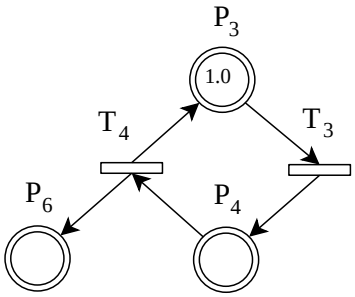

Figure 3.13: The unbounded continuous Petri net.

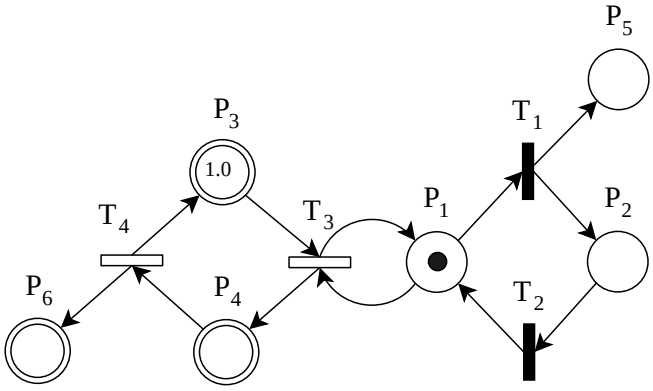

Figure 3.14: The unbounded hybrid Petri net.

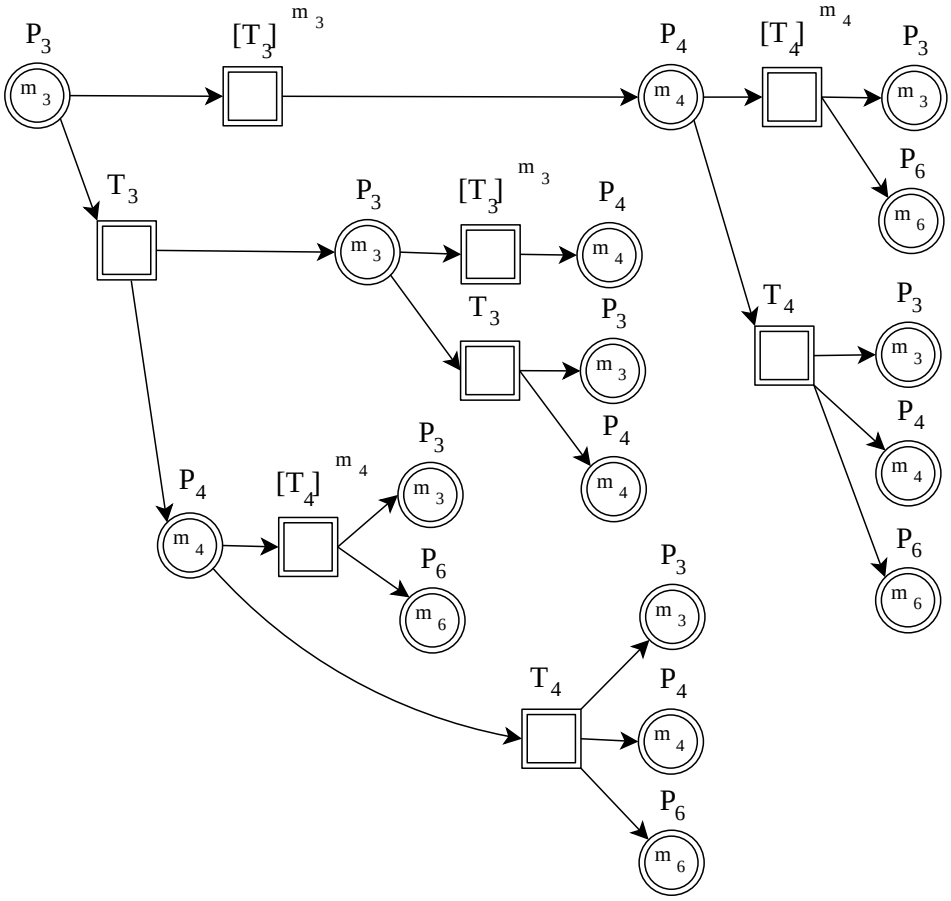

Figure 3.15: The prefix of the unfolding of the unbounded continuous Petri net from Fig. 3.13.

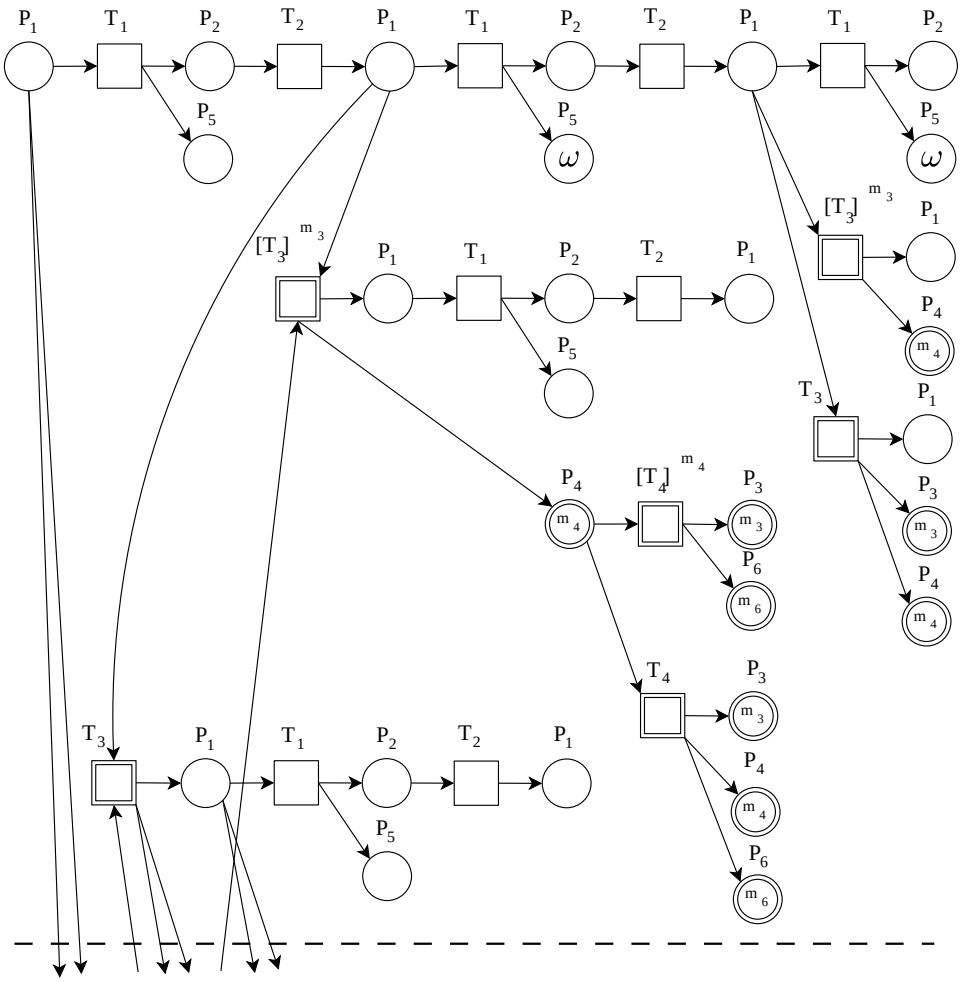

Figure 3.16: The main segment of the finite prefix of the unfolding of the unbounded hybrid Petri net from Fig. 3.14. The whole prefix is not depicted because of size limitations.

Chapter 4

Comparison of Results

The developed algorithms construct the finite net, resp. graph of the continuous and hybrid Petri nets. The algorithms are finite because the number of continuous, resp. discrete macro markings in the continuous, resp. discrete part of the hybrid Petri net is finite. The decision version of the main function is NP-complete.

Each node in the coverability graph represents directly the macro hybrid marking so the whole state space is easier to see. All possible transitions firings between states are explicitly displayed. One can identify deadlocks by finding a node which does not have any outgoing edge.

The unfoldings represents states by cuts which are harder to imagine. On the other hand one can easy see which transitions are sequential, parallel and in conflict. Analysis of the partial order between the transitions occurrences and checking on persistency by analysing the conflicts between the transitions occurrences in the unfoldings is simpler due to absence of cycles.

There exist algorithms for checking qualitative properties such as safeness, boundness, conservativeness and coverability for the hybrid Petri nets from the unfoldings and the coverability graphs. However some information regarding reachability is lost due to the abstraction in the continuous and discrete macro markings in both approaches. The unfoldings and the coverability graphs are over-approximations. Different Petri nets can have the same unfolding, resp. coverability graph. The hybrid Petri net is bounded if and only if the corresponding unfolding, resp. coverability graph does not contain any omega markings. A transition is dead if and only if it does not appear in the corresponding unfolding, resp. coverability graph.

Both approaches can make analysis of the hybrid Petri nets easier. Each of them bring its own view on the state space. The main limit of both approaches is that they may generate huge graphs, resp. nets.

4.1 Example of Scalability

The unfoldings can be smaller than the coverability graphs for the same Petri net especially when the Petri net has a lot of concurrency. Adding independent parallel transitions to a continuous Petri net as shown in Fig.4.1 and Fig.4.4 grows the unfoldings linearly as shown in Fig.4.3 and Fig.4.6 as opposed to growing the coverability graphs as shown Fig.4.2 and Fig.4.2.

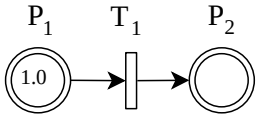

Figure 4.1: The bounded continuous Petri net with one transition.

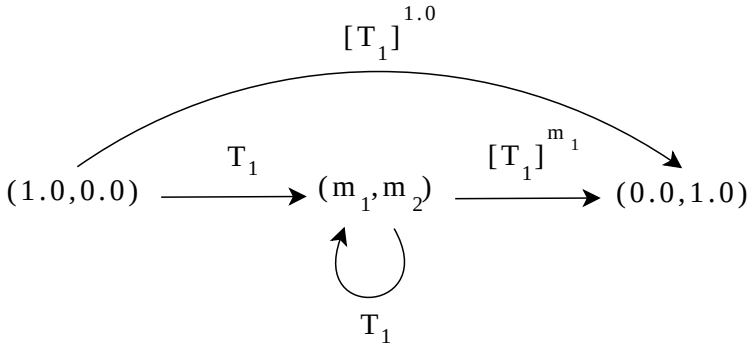

Figure 4.2: The coverability graph of the continuous Petri net in Fig.4.1.

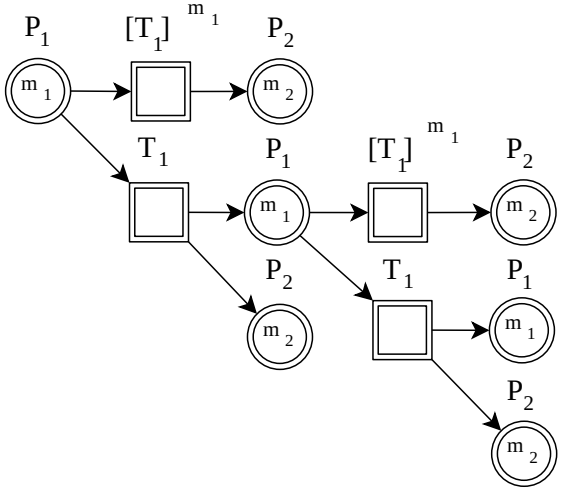

Figure 4.3: The unfoldings of the continuous Petri net in Fig.4.1.

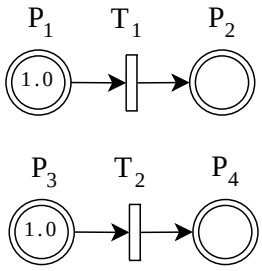

Figure 4.4: The bounded continuous Petri net with two transitions.

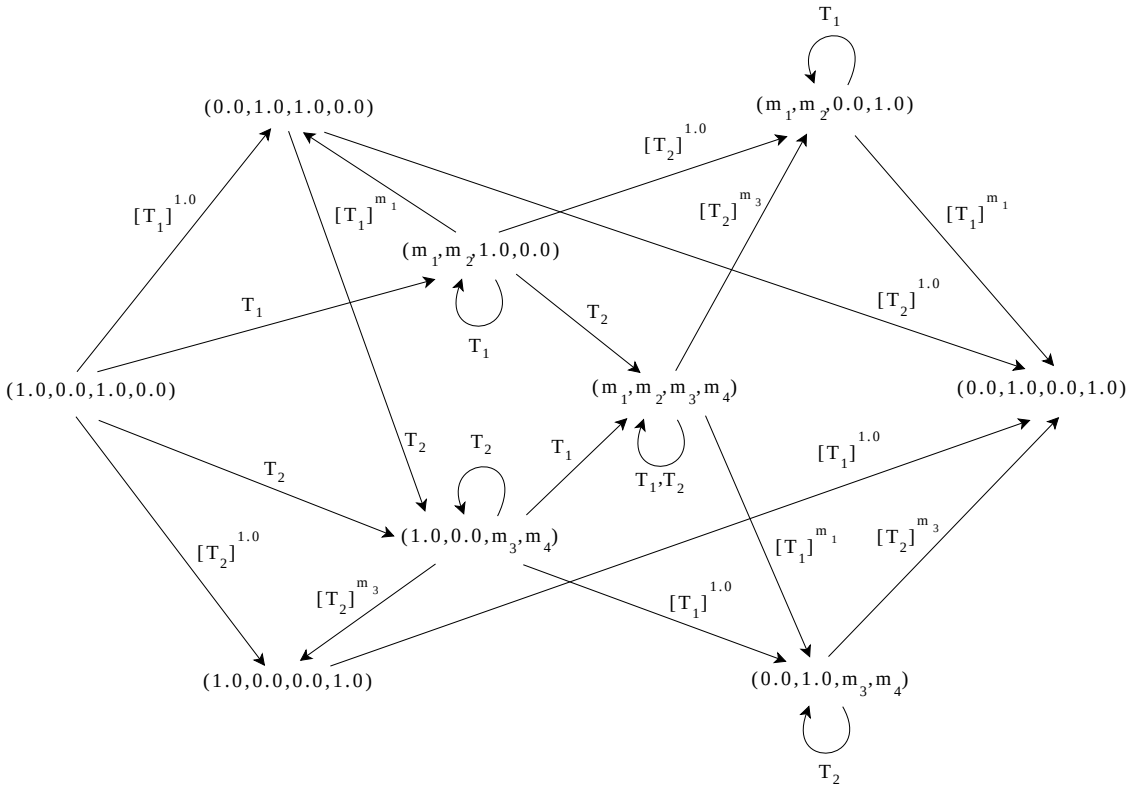

Figure 4.5: The coverability graph of the continuous Petri net in Fig.4.4.

4.2 Example of Hybrid Interaction

Possible discrete transitions firings in Fig.4.7 effectively create a separate continuous coverability sub graph in the resulting coverability graph as shown in Fig.4.8. The influence of the discrete part really works as a 'switch' between two basic behaviors of the continuous part in this example.

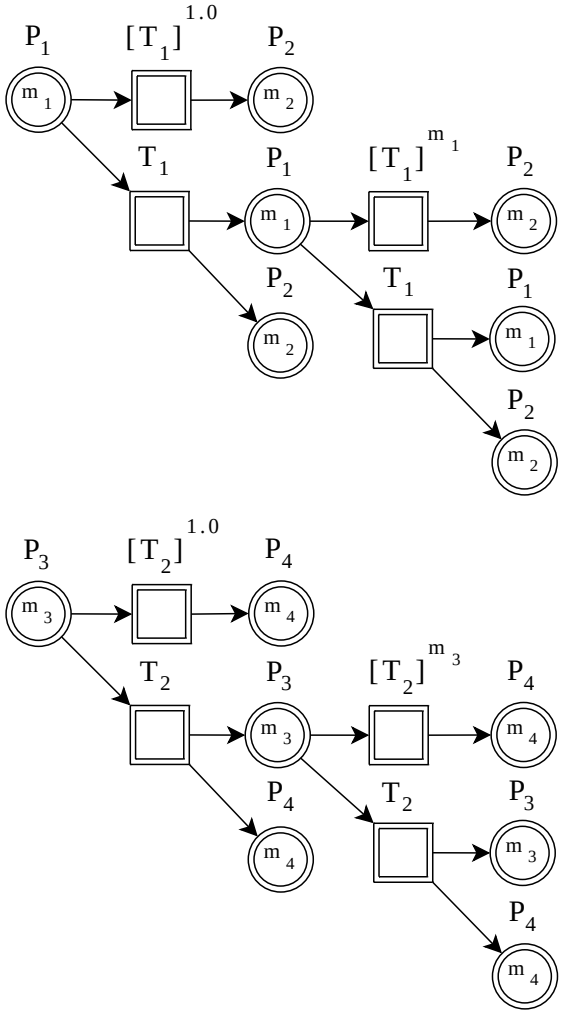

Figure 4.6: The unfoldings of the continuous Petri net in Fig.4.4.

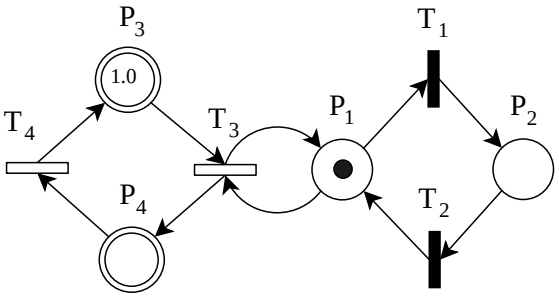

Figure 4.7: The bounded hybrid Petri net.

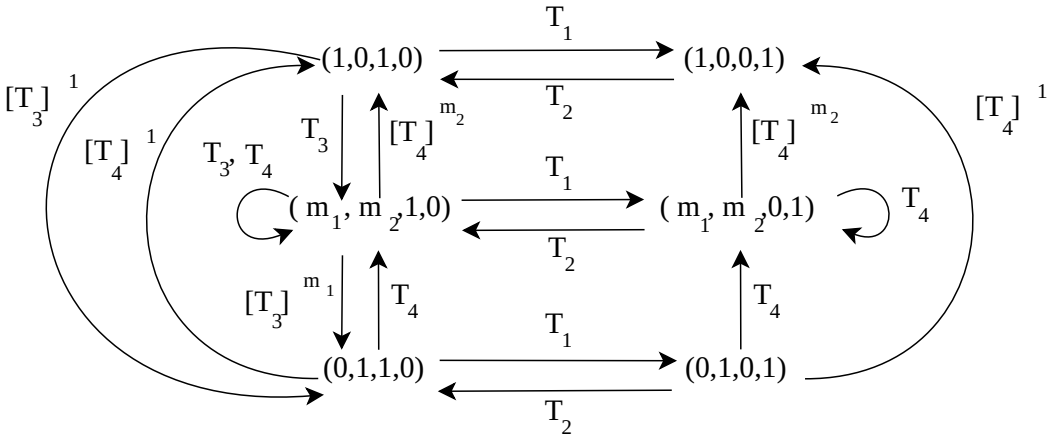

Figure 4.8: The coverability graph of the hybrid Petri net in Fig.4.7.

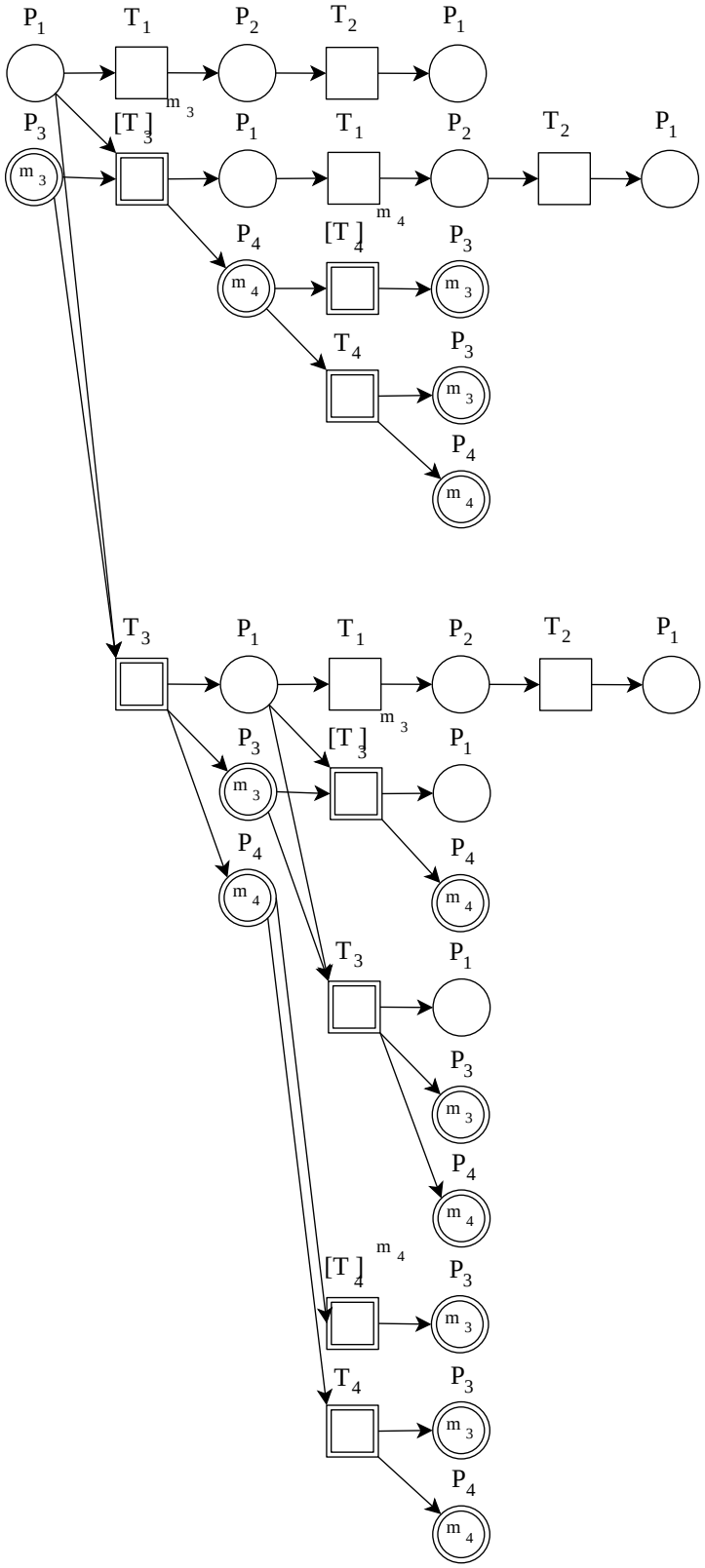

Figure 4.9: The unfoldings of the continuous Petri net in Fig.4.7.

Chapter 5

Conclusions

5.1 Summary

We have presented the formalization of coverability graphs, resp. unfoldings for the hybrid Petri nets together with algorithms for their computation and typical examples.

The coverability graphs and the unfoldings are over-approximations. Some information regarding reachability is lost due to the abstraction in the continuous and discrete macro markings. Nevertheless, both approaches can make analysis of the hybrid Petri nets easier. There exist algorithms for checking qualitative properties such as safeness, boundness, conservativeness and coverability for the hybrid Petri nets from the unfoldings and the coverability graphs. Each of them bring its own view on the state space. Each node in the coverability graph represents directly the macro hybrid marking so the whole state space is easier to see. The unfoldings represents states by cuts which are harder to imagine. The main limit of both approaches is that they may generate huge graphs, resp. nets. Possible firings of independent parallel transitions can grow the unfoldings linearly and the coverability graphs exponentially.

Analysing and verification of countinuous and hybrid Petri nets are one of newly emerging areas in the Petri nets world. We think that this work can serve as a base for additional research in this field.

5.2 Future Work

In the future we plan to implement algorithms for computing coverability graphs and unfoldings for the hybrid Petri nets. And gather some experimental results for a larger hybrid Petri nets.

Bibliography

[1] René David and Hassane Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer Publishing Company, Incorporated, 2nd edition, 2010.

[2] Doi Atsushi, Fujita Sachie, Matsuno Hiroshi, and Nagasaki Masao. Constructing biological pathway models with hybrid functional petri nets. *In Silico Biology*, pages 271–291, 2004.

[3] Rafael S. Costa, Daniel Machado, A. R. Neves, and Susana Vinga. Multi-level dynamic modeling in biological systems - application of hybrid petri nets to network simulation. In *BIOINFORMATICS*, pages 317–321. SciTePress, 2012.

[4] Antti Valmari. The state explosion problem. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the Volumes Are Based on the Advanced Course on Petri Nets*, pages 429–528, London, UK, UK, 1998.

[5] C. Coves, D. Crestani, and F. Prunet. How to manage coverability graphs construction: an overview. In *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, volume 1, pages 541–546 vol.1, 1998.

[6] Jörg Desel, Gabriel Juhás, and Christian Neumair. Finite unfoldings of unbounded petri nets. In Jordi Cortadella and Wolfgang Reisig, editors, *ICATPN*, volume 3099 of *Lecture Notes in Computer Science*, pages 157–176. Springer, 2004.

[7] René David and Hassane Alla. Reachability graph for autonomous continuous petri nets. volume 294 of *Lecture Notes in Control and Information Sciences*, pages 63–70. Springer, 2003.

[8] W. Reisig. *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs in Theoretical Computer Science*. Springer-Verlag, Berlin, 1985.

[9] Jörg Desel, Gabriel Juhás, and Katholische Universitt Eichsttt. What is a petri net? informal answers for the informed reader. In *Unifying Petri Nets, LNCS 2128*, pages 1–27. Springer, 2001.

[10] René David and Hassane Alla. Continuous petri nets. In *Proc. of the 8th European Workshop on Application an Theory of Petri nets*, pages 275–294, Zaragoza, Spain, 1987.

[11] René David and Hassane Alla. Continuous and hybrid petri nets. *Journal of Circuits, Systems, and Computers*, 1998.

[12] Laura Recalde, Enrique Teruel, and Manuel Silva. Autonomous continuous p/t systems. In *Proceedings of the 20th International Conference on Application and Theory of Petri Nets*, pages 107–126, London, UK, UK, 1999.

[13] Richard M. Karp and Raymond E. Miller. Parallel program schemata. *J. Comput. Syst. Sci.*, pages 147–195, May 1969.

[14] Javier Esparza and Keijo Heljanko. *Unfoldings: A Partial-Order Approach to Model Checking (Monographs in Theoretical Computer Science. An EATCS Series)*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[15] Kenneth L. McMillan. A technique of state space search based on unfolding. *Formal Methods in System Design*, pages 45–65, 1995.

[16] Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of mcmillan's unfolding algorithm. *Formal Methods in System Design*, pages 285–310, 2002.

[17] Victor Khomenko and Maciej Koutny. Towards an efficient algorithm for unfolding petri nets. In *Proceedings of the 12th International Conference on Concurrency Theory*, CONCUR '01, pages 366–380, London, UK, UK, 2001.

[18] Matthias Weidlich, Felix Elliger, and Mathias Weske. Generalised computation of behavioural profiles based on petri-net unfoldings. In *Proceedings of the 7th International Conference on Web Services and Formal Methods*, WS-FM'10, pages 101–115, Berlin, Heidelberg, 2011.

[19] Petr Novosad and Milan Češka. Algorithms for computing coverability graphs for continuous petri nets. In *Proceedings of 22th European Simulation and Modelling Conference ESM'2008*, EUROSIS-ETI Publications, pages 489–491, 2008.

[20] Petr Novosad and Milan Češka. Algorithms for computing coverability graphs for hybrid petri nets. In *4th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 177–183. Masaryk University, 2008.

[21] Petr Novosad and Milan Češka. Unfoldings of bounded hybrid petri nets. volume 6927 of *Lecture Notes in Computer Science*, pages 543–550. Springer Berlin Heidelberg, 2012.