ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Διαγράμματα απόφασης πάνω σε Petri Nets

Decision diagrams on Petri Nets

Διπλωματική Εργασία

Μαριέτα Α. Κράντα

Επιβλέποντες Καθηγητές : Μούντανος Ιωάννης
Αναπληρωτής καθηγητής

Donatelli Susanna
Καθηγήτρια

Βόλος, Ιούνιος 2013

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ,
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Διαγράμματα απόφασης πάνω σε Petri Nets

Διπλωματική Εργασία

Μαριέτα Α. Κράντα

Επιβλέποντες : Μούντανος Ιωάννης
Αναπληρωτής καθηγητής

Donatelli Susanna
Καθηγήτρια

Εγκρίθηκε από την διμελή εξεταστική επιτροπή την 28η Ιουνίου 2013

.....	.....
I. Μούντανος	S. Donatelli
Αναπληρωτής Καθηγητής	Καθηγήτρια

Διπλωματική Εργασία για την απόκτηση του Διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας, στα πλαίσια του Προγράμματος Προπτυχιακών Σπουδών του Τμήματος Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας.

……..

Κράντα Μαριέτα

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

Abstract

As computer systems become more pervasive in our everyday life and the number of infrastructure that depends on them increases, simultaneously does the necessity of thorough evaluation and verification of them before the manufacturing phase. For example, a mission critical system (e.g. aviation or automotive braking control) needs to be first well-defined and has its robustness verified through extensive simulations, otherwise a possible error could result in a catastrophic failure. In addition, the necessity for solving increasingly complex problems and coping with the demand of more resources, has led to the implementation of parallel, non deterministic, distributed systems. In order to ease the study and design of such systems, mathematical modeling languages have been constructed. One such language that meets broad acceptance in the scientific community and is widely used in research tools is *Petri Nets*. *Petri Nets* can be used for the systems' evaluation through *Model Checking*; the exhaustive analysis of a system for certain properties and specifications.

In this diploma thesis we focus on the optimization of a performance modeling tool, *GreatSPN*. *GreatSPN* was developed with the collaboration of two Departments : the *Electronic Department of the Technical University of Turin* and the *Department of Informatics of the University of Turin*, in the early '80s. In order to improve execution time, diminish memory footprint, as well as reduce the number of the nodes of the *Decision Diagram* that is created through the generation of its *reachability set*, we exploit a *structural property* of Petri Nets. Furthermore, since different input, results in different complexity, it is of high importance to render the tool user-input independent. In that direction we propose a new heuristic algorithm to approach an optimal solution in a cost efficient manner. The proposed algorithm is evaluated by extending the existing source base of GreatSPN and by studying the behavior of distributed systems per Petri Nets. The systems are described in *Non Stochastic Petri Nets*. Likewise, the properties needed to be evaluated are expressed in *Computational Tree Logic* (CTL), a compatible language to the operation of the tool, for the verification of the necessary system's specifications.

Contents

1 Petri Nets **6**
- 1.1 Introduction . . . 6
- 1.2 Petri Nets Definition . . . 6
- 1.3 Petri Net System Definition . . . 7
- 1.4 Enabling and firing a transition . . . 8
  - 1.4.1 The enabling of a transition without an input inhibitor arcs . . . 8
  - 1.4.2 The firing of a transition . . . 8
  - 1.4.3 The enabling of a transition with an input inhibitor arc . . . 9
- 1.5 State space analysis techniques . . . 9
  - 1.5.1 Reachability Set and Reachability Graph . . . 10
  - 1.5.2 Explicit and Symbolic analysis techniques . . . 11
- 1.6 Properties of the Petri Net systems . . . 12
  - 1.6.1 Behavioral Properties . . . 12
  - 1.6.2 Structural Properties . . . 13
- 1.7 Tool:GreatSPN . . . 14

2 Decision Diagrams **15**
- 2.1 Binary Decision Diagram (BDD) . . . 15
  - 2.1.1 Definition . . . 15
  - 2.1.2 Isomorphic trees, redundant nodes and variable ordering on a Binary Decision Diagram . . . 16
- 2.2 Multi-valued Decision Diagram (MDD) . . . 19
  - 2.2.1 Definition . . . 19
  - 2.2.2 Isomorphic subtrees, Redundant nodes and Variable ordering on a Multi-valued Decision Diagram . . . 19
  - 2.2.3 Encoding a set on an Multi-valued Desicion Diagram . . . 19

3 Verification of the systems **22**
- 3.1 Formal Verification . . . 22
  - 3.1.1 Formal Verification Techniques . . . 22
- 3.2 Model Checking - Verification Technique of GreatSPN . . . 23
  - 3.2.1 Linear Temporal Logic (LTL) . . . 23
  - 3.2.2 Computational Tree Logic (CTL) . . . 23

4 Exploiting P-invariants **26**
4.1 Checking the competitiveness of the tool between the old and the new version of GreatSPN 26
4.1.1 The comparison of the two different versions of the tool 26
4.1.2 The understanding of the results of the comparison between the two version 28
4.2 Reordering of the Places 35
4.2.1 Reordering the places of the three model Petri Net Systems . . . 36
4.3 Proposing a new heuristic algorithm 39

5 Future work **55**

List of Figures

1.1 Bipartite Graph 7
1.2 A Petri Net with an input inhibitor arc on transition T1 7
1.3 A Petri Net with its initial marking 8
1.4 a)Transition T1 is enable b)Transition T1 is not enable, both Petri Nets are with no input inhibitor arcs in T1 9
1.5 a)An immediate transition b) A timed transition 9
1.6 a)The transition T1 is enable b)The transition T1 fires and causes the change of the state of the Petri Net, $M_0[T1\rangle M_1$ 10
1.7 a)Transition T1 is enable b)Transition T1 is not enable, both Petri Nets are with an inhibitor input arc in T1 10
1.8 The read and writers Petri Net system 11
1.9 The reachability graph of the read and writers Petri Net system 12
1.10 A Petri Net with deadlock 14

2.1 Representation of $F(x_1,...,x_i,...,x_n)$ function into a BDD 16
2.2 Merging of two isomorphic subtrees 17
2.3 A reduced BDD representing the function $f = (x_1 \wedge x_2)$ 18
2.4 Representation of $F(S_1,...,S_i,...,S_n)$ function into an MDD 20
2.6 MDD encodes the set **S** 20
2.5 a)Represents the BDD of the function f_1 for the valuable order $x_1>x_3>x_5>x_2>x_4>x_6$ b) Encodes the BDD for the same function with the order $x_1>x_2>x_3>x_4>x_5>x_6$. 21

3.1 a)Xϕ , b)Fϕ, c)Gϕ, d)ψUϕ 24
3.2 a)AXϕ , b)AFϕ, c)AGϕ, d)A(ψUϕ) 24
3.3 a)EXϕ , b)EFϕ, c)EGϕ, d)E(ψUϕ) 25

4.1 The Kanban model 27
4.2 The Flexible Manufacturing System 28
4.3 The MAPK model 28
4.4 Petri Net consisted of three independent Petri Nets 36
4.5 Petri Net which consists of two shared places 37
4.6 Fork and join Petri Net 39

List of Tables

4.1 Non safety Property for Kanban Model- New version of the tool 29
4.2 Non safety Property for Kanban Model- Old version of the tool 29
4.3 Deadlock Property for Kanban Model- New version of the tool 30
4.4 Deadlock Property for Kanban Model- Old version of the tool 30
4.5 Reversibility Property for Kanban Model- New version of the tool 31
4.6 Reversibility Property for Kanban Model- Old version of the tool 31
4.7 Non Safety Property for FMS Model- New version of the tool 31
4.8 Non Safety Property for FMS Model- Old version of the tool 32
4.9 Deadlock Property for FMS Model- New version of the tool 32
4.10 Deadlock Property for FMS Model- New version of the tool 32
4.11 Reversibility Property for FMS Model- New version of the tool 33
4.12 Reversibility Property for FMS Model- Old version of the tool 33
4.13 Non Safety Property for MAPK Model- New version of the tool 33
4.14 Non Safety Property for MAPK Model- Old version of the tool 34
4.15 Deadlock Property for MAPK Model- New version of the tool 34
4.16 Deadlock Property for MAPK Model- Old version of the tool 34
4.17 Reversibility Property for MAPK Model- New version of the tool 35
4.18 Reversibility Property for MAPK Model- Old version of the tool 35
4.19 First Model (Places = 9 & Bound = 1) 36
4.20 Second Model (Places = 11 & Bound = 3) 38
4.21 Second Model (Places = 11 & Bound = 3) 38
4.22 Third Model (Places = 6 & Bound =5) 39
4.23 FMS model without the evaluation of a CTL property- New version of the tool without and with the new heuristic algorithm 42
4.24 FMS model evaluating the property- AF deadlock- New version of the tool without and with the new heuristic algorithm 43
4.25 FMS model evaluating the property- AG(M1=0→(P1wM1)<N) - New version of the tool without and with the new heuristic algorithm 44
4.26 FMS model evaluating the property- AG(M2=0→(P3M2+P2wM2)<7) - New version of the tool without and with the new heuristic algorithm . . 45
4.27 FMS model evaluating the property- EF(M1!=0 and not en(tM1)) - New version of the tool without and with the new heuristic algorithm 46
4.28 FMS model evaluating the property- EG not (P1s=N and P2s=N and P3s=N) - New version of the tool without and with the new heuristic algorithm 47

4.29 FMS model evaluating the property- E[(M1>0)U(P1s=N and P2s=N and P3s=N)]-New version of the tool without and with the new heuristic algorithm 48
4.30 FMS model evaluating the non safety property - New version of the tool without and with the new heuristic algorithm 49
4.31 FMS model evaluating the deadlock property - New version of the tool without and with the new heuristic algorithm 50
4.32 FMS model evaluating the reversibility property - New version of the tool without and with the new heuristic algorithm 51
4.33 Kanban model evaluating the non safety property - New version of the tool without and with the new heuristic algorithm 52
4.34 Kanban model evaluating the deadlock property - New version of the tool without and with the new heuristic algorithm 53
4.35 Kanban model evaluating the reversibility property - New version of the tool without and with the new heuristic algorithm 54

Chapter 1

Petri Nets

1.1 Introduction

As the problems we try to solve become more complex and demand for more processing power and resources, the computers that are used to solve them have evolved from sequential computing machines to parallel, distributed, non-deterministic systems. Their data flow depends entirely on the events that occur , synchronous or not, in order to change their current state. Systems that depend on the condition of a state and on the events that will change it are called *Discrete Events Dynamic Systems* (DEDS). Mathematical modeling languages have been introduced in order to ease the study of DEDS systems; one such language [2] is *Petri Nets*. Petri Nets [14]can provide a precise description of complex systems and also by exploiting their structure, more information for the behavior of the system is supplied, i.e. the existence of a deadlock. An additional advantage is that the analysis techniques of Petri Nets depend on the user's preferences and skills. Petri Nets were first introduced by *Carl Adam Petri* on his Ph.D thesis[19], [20]. Different types of Petri Nets have been introduced to help with the analysis demands of diverse systems. *Stochastic Petri Nets* [18] are Petri Nets where the delays of the firing of the transitions are random variables according to a negative exponential probability density function. *Generalized Stochastic Petri Nets* are similar to the Stochastic Petri Nets with the addition of the existence of immediate transitions (transition with zero firing delay).*Ordinary Petri Nets* are characterized by the multiplicity of the input/output arrows that can only be zero or one. *Colored Petri Nets*[18] permit the existence of distinguished tokens, the diversity of which is declared by a color; each color represents a group of any kind of information. In this thesis we focus on the study of *Non Stochastic Petri Nets*.

1.2 Petri Nets Definition

Petri Nets are based on a bipartite graph; a graph which consists of two disjoint sets ($P \cap T = \emptyset$) and each edge can connect an element from one set to an element from the other set. Two elements from the same set can not be connected and odd circles are not permitted. More formally a Petri Net is a six tuple $\{ P, T, F, W, H \}$ where :

- P symbolizes the places of a Petri Net;i.e the one of the disjoint sets and each place

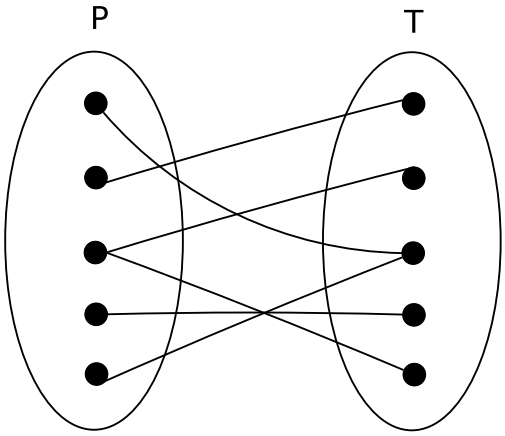

Figure 1.1: Bipartite Graph

represents the condition of the system and are illustrated by circles

- T symbolizes the transitions of a Petri Net; i.e the second set. Each transition represents the events that change the state of the system and are illustrated by rectangles
- F $\subseteq$ (P$\times$T)$\cup$(T$\times$P) is the set of directed arcs that connect each place to a transition and vice versa
- W the function that assigns to each arc an integer which declares the multiplicity or the weight of each arc, W: F $\rightarrow \mathbb{N}$, an arc without an assigned number is considered as the multiplicity of the arc is 1
- H symbolizes the inhibitor arcs. Inhibitors are illustrated with an arc, at the end of which there is a circle.

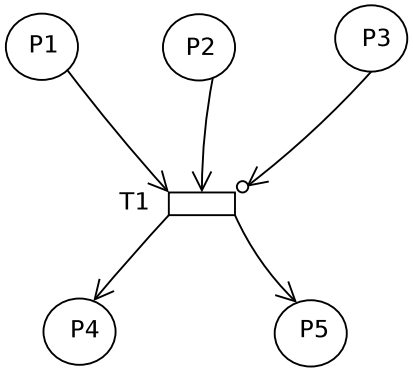

Figure 1.2: A Petri Net with an input inhibitor arc on transition T1

1.3 Petri Net System Definition

In order to describe the activities of the system on a Petri Net it is mandatory to represent them in a graphical way too. Each activity is represented by *a token*, depicted

by a dot. The function that assigns in each place a positive number of tokens is called *Marking*. Graphically the tokens are dots that are contained inside the circle that depicts the place.

$$m : P \rightarrow \mathbb{N}^{+}$$

A Petri Net with a fixed initial marking is called a Petri Net system [14]. Summing the two definitions above, a Petri Net system is characterized by its static structure and by its initial marking M_0; the marking that is assigned on the initial state of the system. So a more complete definition of a Petri Net is

$$\{ P, T, F, W, H, M_0 \}$$

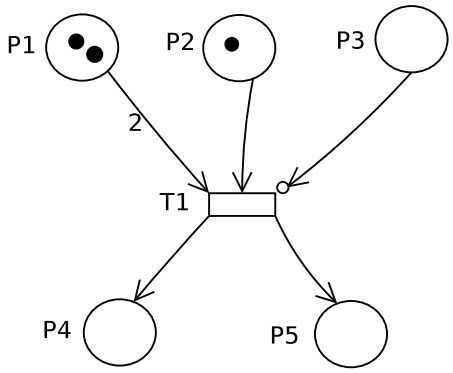

Figure 1.3: A Petri Net with its initial marking

1.4 Enabling and firing a transition

1.4.1 The enabling of a transition without an input inhibitor arcs

In order to study the different states that a Petri Net can meet it is necessary to define the role of the transitions that cause the state changes. A transition is enabled if and only if the number of tokens of the incoming places is equal or greater than the weight/multiplicity of the incoming arcs,that is:

$$\forall p \in P: M(p) \geq W(p,t), \text{where } t \in T$$

1.4.2 The firing of a transition

The firing of a transition tends to change the state of a Petri Net. The firing causes the removal of the number of tokens ,that is assigned at the input arc, from the input places and adds the number of token assigned in the output arc of the transition in the output place. There are two different types of transitions: the immediate transitions that fire with zero delay and the timed transitions which on *Stochastic Petri Net systems* fire with a negative exponential delay. The transitions are depicted as shown on *Figures 1.5, 1.6.*

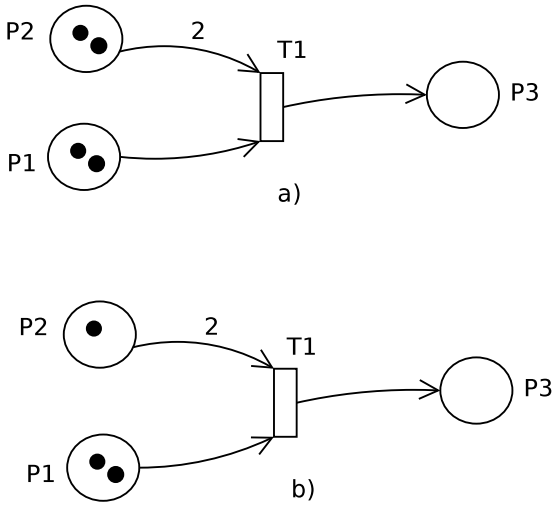

Figure 1.4: a)Transition T1 is enable b)Transition T1 is not enable, both Petri Nets are with no input inhibitor arcs in T1

Figure 1.5: a)An immediate transition b) A timed transition

1.4.3 The enabling of a transition with an input inhibitor arc

By default an inhibitor arc, as its name declares, searches for *no tokens* in a place, thus the enabling of a transition with an input inhibitor arc has an additional rule. More specifically, a transition is enabled if and only if the number of tokens of the incoming places is equal to or greater than the weight/multiplicity of the incoming arcs, and the number of tokens of the incoming place is less than the weight/multiplicity of the incoming inhibitor arc in a transition , i.e.

$$\forall p \in P : M(p) \geq W(p,t), \text{where } t \in T \textbf{ AND } M(p) \lneq H(p,t)$$

The firing of a transition independently of the type of the input arcs evokes the exact same movements of tokens. Since the behavior of a Petri Net is nondeterministic, multiple different transitions may be enable at the same time, from which any one may fire. The sequence of transitions that fire $\{\sigma\}$ in order to approach all the reachable states is not known a priori.

1.5 State space analysis techniques

There are numerous analysis techniques for Petri Nets,each one more suitable for systems that comply with the definition of different type of Petri Nets. In this thesis we choose to concentrate on reachability analysis techniques. Reachability analysis techniques are

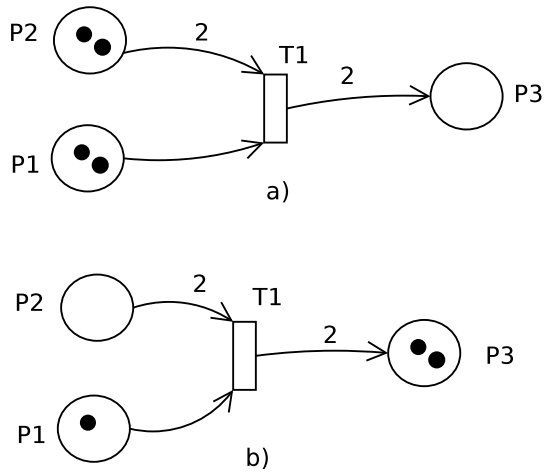

Figure 1.6: a)The transition T1 is enable b)The transition T1 fires and causes the change of the state of the Petri Net, $M_0[T1\rangle M_1$

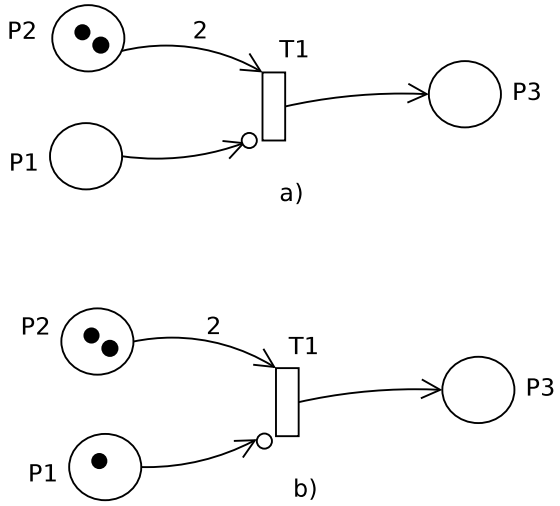

Figure 1.7: a)Transition T1 is enable b)Transition T1 is not enable, both Petri Nets are with an inhibitor input arc in T1

based on the reachability set of a system and the information provided by the reachability graph on both of which we will elaborate in the next paragraphs.This specific technique is [14] considered strong since it provides the complete behavior of a system through the reachability set and reachability graph.

1.5.1 Reachability Set and Reachability Graph

Reachability Set (RS) Given an initial marking of a Petri Net system, M_0, the reachability set is denoted as *RS(M_0)* and it denotes the smallest set of markings on a given Petri Nets system such that

$$M_0 \in RS(M_0)\ M_1 \in RS(M_0) \wedge \exists t \in T\colon M_1[t\rangle M_2 \Rightarrow M_2 \in RS(M_0)$$

The RS is built incrementally starting from M_0, the initial marking, and after the firing of a transition or a set of transitions a new marking M_1 is added on the RS(M_0). Iteratively every new marking after the transitions'/transition's firings is added on the RS, creating the reachability set of the system. Once a marking M_i is already in RS it must not be included in successive iterations. The reachability set contains no information about the firing of the transitions that change the state of a system, which leads to the new marking. The algorithm that builds the RS terminates if the RS is finite.

Reachability Graph (RG) The Reachability Graph as its name declares is a labeled directed graph whose set of nodes is RS, and the set of arcs declares the transitions that cause the change of state of a Petri Net system and lead to each new marking. More formally the set of arcs denotes

$$\mathrm{A} \subseteq \mathrm{RS} \times \mathrm{RS} \times \mathrm{T} < M_i, M_j, t > \Leftrightarrow M_i[t\rangle M_j$$

At each iteration and for each reachable marking the algorithm contracting the reachability graph adds one arc labeled with the transition, in the set A. When the reachability set is finite, then the reachability graph is also finite. *Figure 1.8* below, for K=2 and initial marking M_0={2p1+p5}, shows the reachability graph of the readers and writers Petri Net system [6]. The nodes declare the reachability set, all reachable markings from the initial marking M_0, and the arcs are labeled with the transition that leads from one marking to another.We borrow the two following *Figures 1.8 and 1.9* from [14] so as to show the reachability graph from a larger that a 5-place Petri Net system, which we have not seen so far.

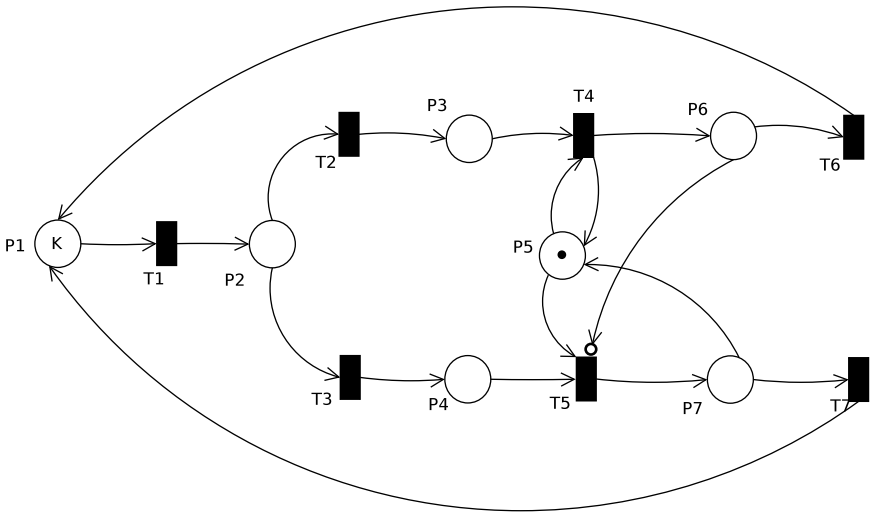

Figure 1.8: The read and writers Petri Net system

1.5.2 Explicit and Symbolic analysis techniques

The *Explicit generation* technique[9] of the reachability set of a system adds in each step a new state. The disadvantage of this analysis technique is that the memory

footprint increases linearly to the number of states. To avoid the linear increment of memory footprint another more "proper" technique is introduced,the *symbolic generation* technique. Symbolic analysis adds set of states to the reachability set, and that is how the word "proper" is justified since the Petri Net systems we focus on the analysis of the reachability set consist of sets of places in each marking. In this direction we will introduce in detail *Decision Diagrams*, which are used to represent in a compact way the reachability set for the study/examination of the systems.

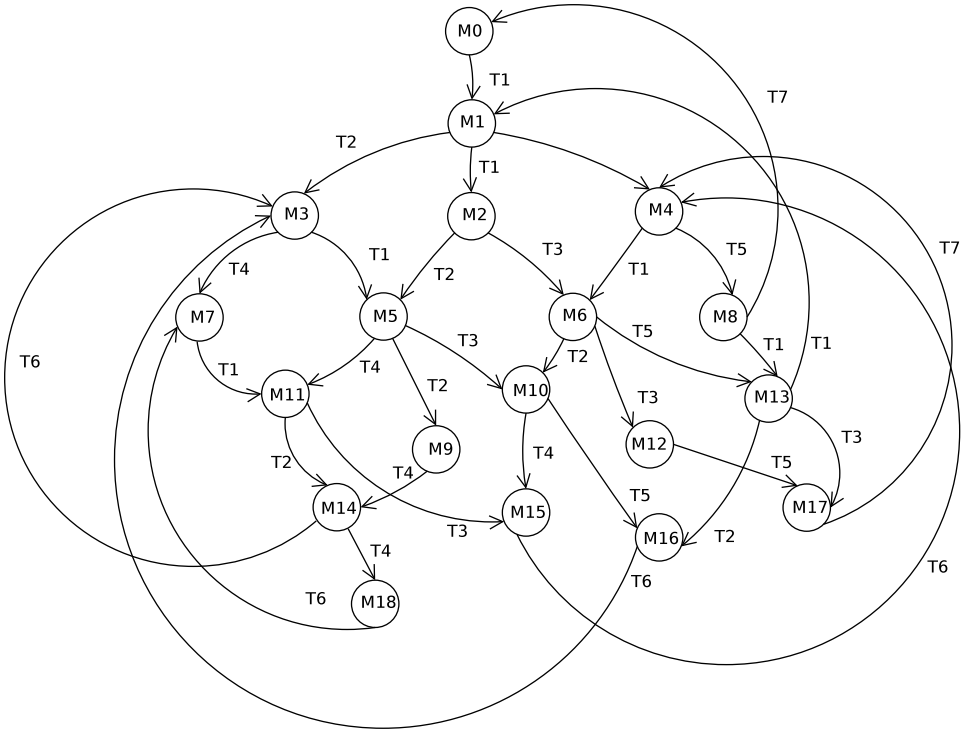

Figure 1.9: The reachability graph of the read and writers Petri Net system

1.6 Properties of the Petri Net systems

Introduction Petri Nets offer a number of properties which are associated with concurrent, parallel systems. There are two type [16] of properties, the *Behavioral Properties* and the *Structural Properties*. The first type of properties depend entirely on the initial marking of the Petri Net, whilst the second ones depend only on the structure of the Petri Net.

1.6.1 Behavioral Properties

There are some basic behavioral properties [16],[14] which aid in the analysis of the system and the determination of its specifications. These are *Reachability*, *Boundness*, *Deadlock* and *Reversibility*. A detailed description of each property follows.

Reachability A marking M_i is said to be reachable by another marking M_{i-1} after the firing of a transition t_i or after the firing of a sequence of transitions $\{\sigma\}$. The reachable marking is denoted by M_{i-1} [t $\rangle M_i$ and $M_i[\{\sigma\}\rangle M_i$, respectively. The reachable marking declares the new state of the system. We can use this properties to discover if the Petri Net will ever be in a desired or undesired state. Let us take for example the Petri Net in *Figure 1.5a*. In this case we only have one transition that can fire. The initial marking of the system is $M_0 = \{$ 2p1 + 2p2$\}$and as we have already said the transition is enabled so it may fire. After the firing of the transition T1 the system is transformed to its new state that is shown in *Figure 1.5b*. The new reachable marking is $M_1 = \{$ 1p1 + 2p3 $\}$; i.e. according to what we have already said M_0 [T1 $\rangle M_1$.

Boundness A Petri Net is said to be *k-bounded* if the number of tokens in all places is less than or equal to the finite number denoted by k. For every arbitrary sequence of transitions firings the number of tokens must not exceed the number k in all reachable markings. A Petri Net that is 1-bounded is called a safe Petri Net. A k-bounded Petri Net guarantees a finite reachability set which ease its study, since the new states of the net are not infinite. In the present thesis we focus on the study of bounded Petri Nets.

Deadlock A Petri Net has a deadlock if and only if there is a subset of places $P_i \subseteq P$ that once they become unmarked, they stay unmarked for the rest of the execution time. In particular, a deadlock is the subset of places such that the set of its input transitions is a subset of the set of its output transitions. *Figure 1.10* shows a Petri Net that has deadlock. As we can see both transitions {T1,T2}, since they are enabled, fire, and the input places {P2,P3} become unmarked. Thus, due to the structure of the system none of the transitions will be enabled again and the Petri Net has a deadlock. It is clear that the set of its input transitions (T1) is a subset of the set of its output transitions (T1,T2).

Reversibility A Petri Net is reversible if and only if from any marking, $m \in RS(M_0)$ the initial marking is reachable from m. This means that from every reachable marking we can always get back to the initial state of the Petri Net. A Petri Net is reversible if and only if

$$M_i \in RS(M_0) \text{ and } M_0 \in RS(M_i).$$

1.6.2 Structural Properties

As was stated earlier a structural property depends only on the structure of a Petri Net. Two basic structural properties are *Place-invariant* (P-invariant) and *Transition-invariant* (T-invariant). As an invariant we denote an assertion that is true in all the reachable states of a system.

Place-invariant A P-invariant is an independent variable that demonstrates a set of places in which the sum of the tokens is stable independently of the firing sequence $\{\sigma$ $\}$ of the transitions.

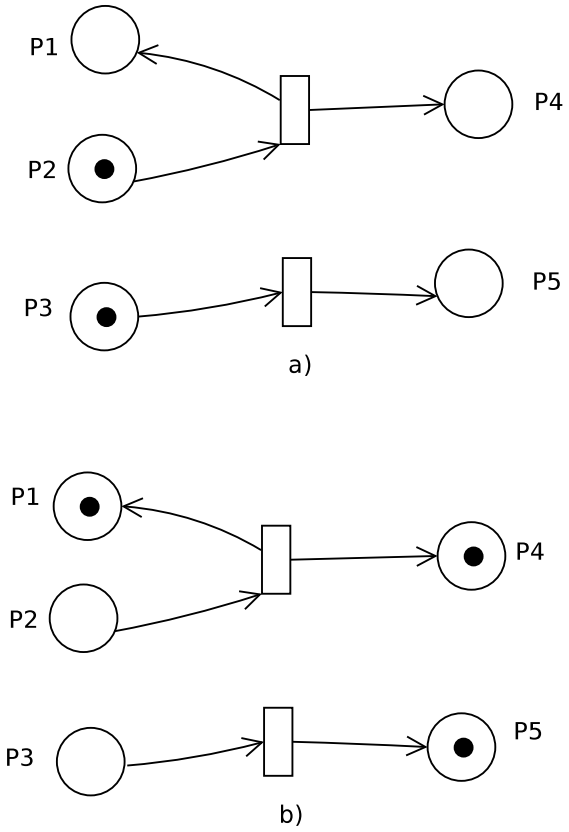

Figure 1.10: A Petri Net with deadlock

Transition-invariant A T-invariant is a variable that demonstrates a set of the transitions after the firing in which we end up on the initial marking/state of the system.

Structural properties will play a crucial role on this diploma thesis as will be demonstrated below. Before we proceed on the work that was implemented on this thesis it is necessary to make the reader familiar with some formal definitions that are useful for the comprehension of our contribution.

1.7 Tool:GreatSPN

GreatSPN is a performance modeling tool that was built in Turin in the early '80s. The tool is the result of the collaboration of two Universities, the University of Turin and the Technical University of Turin [18]. GreatSPN offers a wide variety of solvers for the study of Petri Nets that we have already mentioned. In addition it offers a graphical user interface (GUI) through which Petri Nets are built graphically. All the experiments that are described later on were implemented solely on the GreatSPN tool.

Chapter 2

Decision Diagrams

Introduction

Decision diagrams [11] are directed acyclic graphs which consist of two types of nodes, terminal and non terminal nodes, and edges. A Decision Diagram encodes the type of function

$$f\colon (S_1 \times S_2 \times ... \times S_n) \rightarrow \{0.1\}$$

The sets S_{is} are called *Domains* and are defined over state variables of the system (x_1,...,x_n) , and each edge, labeled by a possible value, leads from one state of the system to the other. Every non terminal node can have as many output edges as the range of the values of the state variables, though the nodes can't have duplicate outputs. The terminal nodes can be numbered with {0,1}, which declares the value of the represented function. Starting from the root of the diagram and following the edges to the terminal nodes,through a path that the numerical input assert, we resolve the function to its final result.

A Decision Diagram is referred to as a *Binary Decision Diagram* (BDD) if and only if all the domains are equivalent $S_1 = S_2 = ... = S_n$ and the ascertained number of its state variables is $x_1 = x_2 = ... = x_n = \{0,1\}$, accrediting to a digital function.

On the contrary, we refer to a *Multi-valued Decision Diagram* (MDD) when the maximum scope of each domain value is {0,1,...,n}. Note that in MDDs each domain may conclude a different range of values. Hence, each domain is expressible by state variable $S_i = \{x_i\}$ and each state variable assumes a finite number of values $x_i = \{0,1,...,k\text{-}1\}$. From now on we will refer to the state variable of decision diagram as a node, in order to focus on the physical structure of the diagram rather than the mathematical one.

2.1 Binary Decision Diagram (BDD)

2.1.1 Definition

Binary Decision Diagrams were first introduced by Aker [3].

A Binary Decision Diagram:

- consists of two types of nodes, *terminal* and *non terminal*
- the range of the values for each node is 0,1
- the terminal nodes are in 0 level, and the non terminal are among $max\_level \geq non\ terminal\_nodes\_level \geq 1$
- for each possible assigned value each node has an outgoing edge pointing to the node's child, in the next-lower level of the diagram.
- each node at a level encodes the **Shannon expansion theorem** [22]

$$F(x_1,...,x_i,...,x_n) = \bar{x}_i * F(...,x_i=0,...) + x_i * F(...,x_i=1,...)$$

the function assigns to each variable, recursively, both the valid values, in order to obtain the function's $F(x_1,...,x_i,...,x_n)$ numerical result.

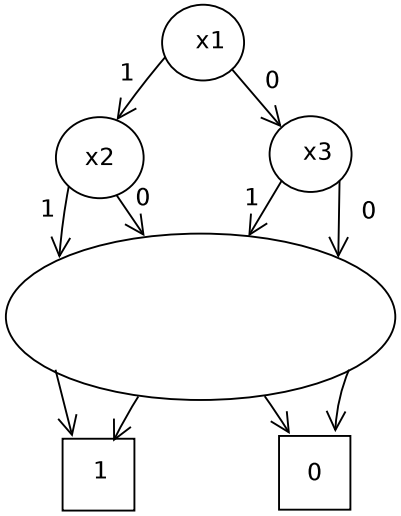

Figure 2.1: Representation of $F(x_1,...,x_i,...,x_n)$ function into a BDD

In consideration of making the diagrams easier for the readers, instead of declaring the value of each edge in the the diagram we will represent the 0 value with a **dashed arrow** and the value 1 with a **consecutive arrow**.

2.1.2 Isomorphic trees, redundant nodes and variable ordering on a Binary Decision Diagram

The number of the nodes of a BDD, i.e its size, representing a function **F** is growing exponentially [3] with the number of the variables. In the worst case the diagram will have $2^n - 1$ nodes. From this point of view, the use of a BDD does not seem useful at all, since for functions with more than 5 variables, the size of its representing diagram will be large enough to be manipulated efficiently. For example, the function

$F(x_1,...,x_8) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee (x_5 \wedge x_6) \vee (x_7 \wedge x_8)$ needs $2^8 - 1 = \mathbf{255}$ nodes

A number which declares no significant difference to the size of a Karnaugh map or a truth table representing the same function. Nevertheless, the size of a binary decision diagram is changeable
alterable.By taking advantage of the isomorphic trees and the redundant nodes that appear in the graphical representation we can create smaller decision diagrams. As well, also variable order affects the size of the decision diagrams.

Isomorphic subtrees Two subtrees on a binary decision diagram are called isomorphic [11] if the set of nodes (S_1,S_2) in each tree encodes the same function, $f^{S_1} = f^{S_2}$. Two isomorphic trees have exactly the same structure and end up in the same terminal nodes. For this reason we can merge them into one subtree and traverse the edge from the previous node of one isomorphic to the remaining one in the diagram.

As we can see in the *Figure 2.2* on the left image four subtrees compose the binary decision diagram that encodes a function *f*, two of which, the red ones are isomorphic ($f^{S_1} = f^{S_2}$). After their merging the final diagram we acquire is the one in the right side of the image.A procedure that leads to a smaller BDD, a *Merged* BDD.

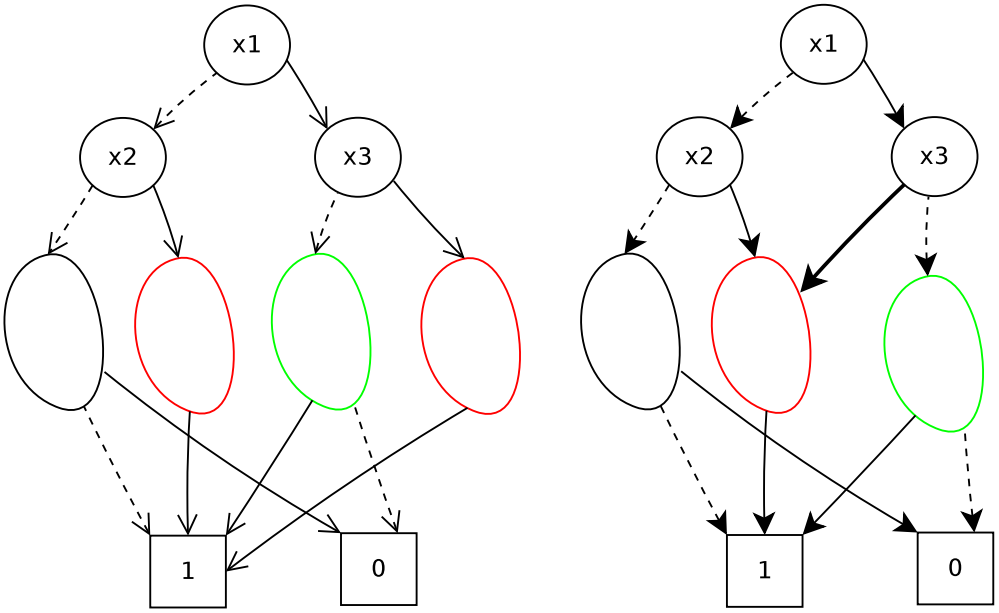

Figure 2.2: Merging of two isomorphic subtrees

Redundant nodes Redundant nodes [11] are called the non terminal ones that their descriptive function doesn't depend on the value of its variable and the value of its descendant nodes. The output of the path that is led from a redundant node x_i is independent of the values of the nodes (x_{i+1},...,x_n). The redundant nodes can be eliminated since no new information is provided for the function's interpretation.

The *Figure 2.3* shows the function that describes the *AND* gate,

$$f = x_1 \wedge x_2$$

As we observe the output of the function if $x_1 = 0$ doesn't depend on the value of x_2. Any value that can be assigned in the variable x_2 doesn't change the numerical result of *f*. In this case we can eliminate the left x_2 node and connect directly x_1 to the terminal node 0, as it is shown on the example below.After the nodes' elimination we manage to create a smaller BDD, a *Reduced* BDD.

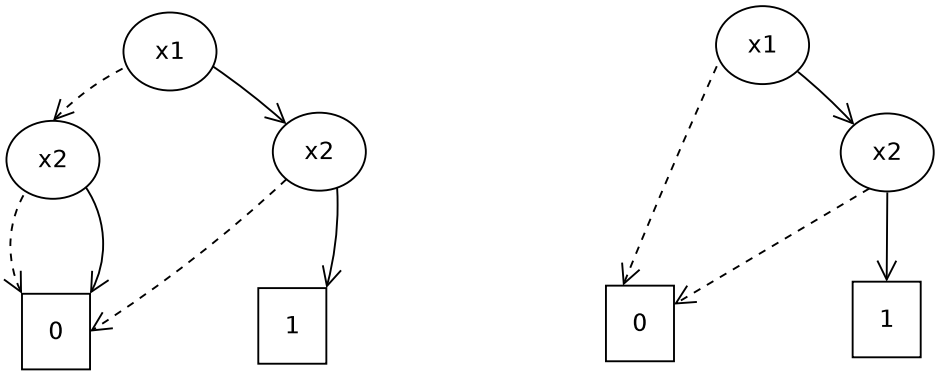

Figure 2.3: A reduced BDD representing the function $f = (x_1 \wedge x_2)$

Variable ordering The ordering of the variables,i.e. in which level of the decision diagram each variable appears, is also crucial for the diagram's size. We can observe that different variable ordering can lead to a smaller or a bigger BDD. The *Figure 2.4* shows the BDD representation of the function

$$f_1 = (x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee (x_5 \wedge x_6)$$

for two different variable orderings. It is noticeable that even though the function consists of 6 variables, with random variable ordering we can accumulate a decision diagram with 14 nodes as the left image of the figure shows. Nonetheless, we can create a BDD with less than a half of the nodes with a totally different ordering as the right image shows.

In order to make the use of the BDDs efficient it is important to approach the best variable ordering as far as the number of nodes that consists the diagram is concerned. However, finding the optimal variable ordering is an NP-complete problem [7]. Hence, it is important to approach the the best ordering through heuristic algorithms. As the number of variables grows, so does the complexity for discovering a good variable order.

2.2 Multi-valued Decision Diagram (MDD)

2.2.1 Definition

Multi-valued Decision Diagrams are a natural extension of Binary Decision Diagrams. The need for extending the structure and the rules that govern the Binary Decision Diagrams, is for describing more complex functions, where the scope of numeric assignment on the variables or the set of variables is : {0,1,...,k-1}. MDD is a directed acyclic graph [9] which encodes a function f : {0,1,...,k-1}$^L \rightarrow$ {0,1}.

More precise:

- consists of two types of nodes, *terminal* and *non terminal*
- the range of the values for terminal nodes is 0,1
- non terminal nodes can be evaluated among 0,1,...,k-1
- the terminal nodes are in 0 level, and the non terminal are among $max\_level \geq non\ terminal\_nodes\_level \geq 1$
- for each possible assigned value each node has an outgoing edge pointing to the node's child, in the next level of the diagram.
- each node at a level encodes the characteristic function υ_p, recursively

$$\upsilon_p(x_1, ..., x_n) = \begin{cases} p & level = 0 \\ \upsilon_{p[xi}(x_1, ..., x_n) & level > 0 \end{cases}$$

2.2.2 Isomorphic subtrees, Redundant nodes and Variable ordering on a Multi-valued Decision Diagram

Since MDDs are a natural extension of BDDs, MDDs inherit [11] the above properties that were just described for binary decision diagrams. The only difference with the BDDs is that we obtain smaller MDDs by exploiting the isomorphic trees and not by exploiting the redundant nodes. It is proved [10] that by eliminating the nodes on this type of decision diagrams it is not guaranteed that we will receive an MDD with a smaller number of nodes. However, the merge of isomorphic subtrees guarantees a smaller MDD. In Petri Nets, and specifically on GreatSPN [8],[18] tool we use the MDDs to encode the reachability set of the Petri Net systems.

2.2.3 Encoding a set on an Multi-valued Desicion Diagram

In consideration of understanding the representation of a set into an MDD, it is obligatory to do so with an example. We can assume that the set to be expressed in a decision diagram is shown on the *Figure 2.7*. From the definition of the decision diagrams, it is stated that the expression of the function is : $S_1 \times ... \times S_n$. As we can observe in the

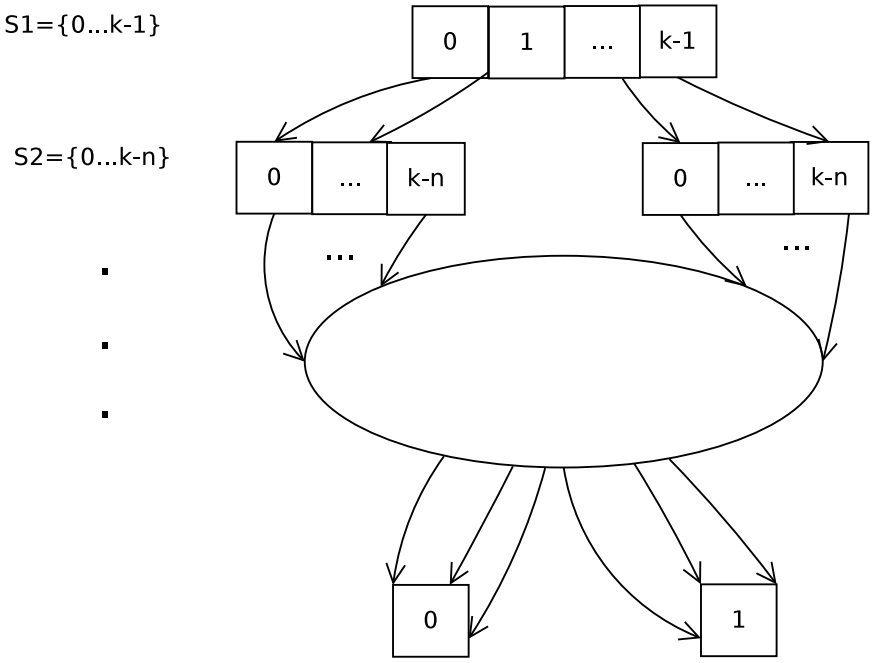

Figure 2.4: Representation of F(S_1,...,S_i,...,S_n) function into an MDD

example given we have three domains where each one of them declares a state variable. $S_1 = x_1$, $S_2 = x_2$ and finally $S_3 = x_3$. Each state variable is covered by a finite number of values, x_1={0,1,2}, x_2={0,1,2} and x_3={0,1}.

The order of the domain plays a crucial role on the size of the MDD. For the example we consider the order x_1¿x_2¿x_3. We create the MDD top-bottom. Starting for each value of x_1 we continue to the next domain's valued, guided by the known states that the function recognizes, and we recursively, apply the characteristic function, we formerly presented, to built the decision diagram.Each output edge from a node connects the whole next domain not only the node of the domain that the arrows are shown that they are attached to.The set of states that the decision diagram in *Figure 2.6* recognizes is *S*={*000 001 010 011 020 021 101 121 210 220*}

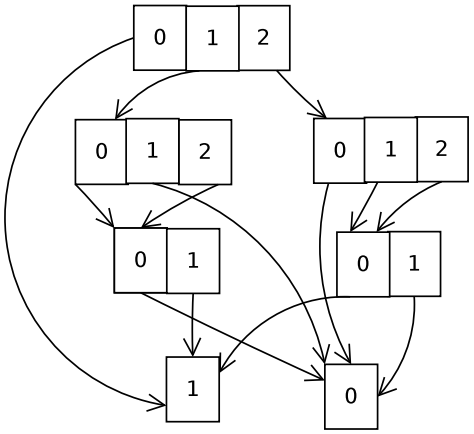

Figure 2.6: MDD encodes the set **S**

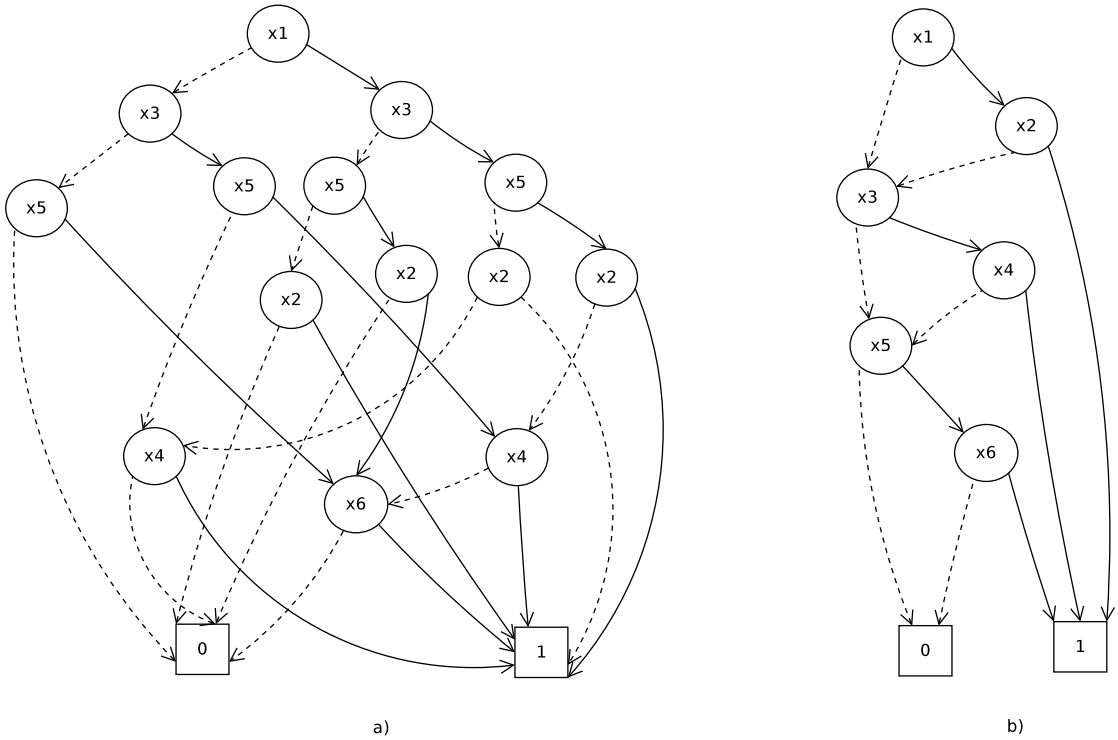

Figure 2.5: a)Represents the BDD of the function f_1 for the valuable order $x_1>x_3>x_5>x_2>x_4>x_6$ b) Encodes the BDD for the same function with the order $x_1>x_2>x_3>x_4>x_5>x_6$.

Specifically, in the tool GreatSPN in order to observe the behavior of a Petri Net we choose the *symbolic* state-space analysis as we have previously declared. We encode the system's *reachability set* into an L-level MDD. L declares the number of levels,i.e the number of domains in the constructed MDD. Each domain on the MDD contains all the possible reachable markings that the Petri Net meets. Each marking is a state variable and the number of places that appear in the Petri Net declares the maximum value of the state variables. The function that describes the $\{\sigma\}$ sequence of transition firing is also encoded on an MDD. Particularly, in a 2L-level MDD. The number of levels needed to characterize the diagram is double due to the description of the *Next-state function* that this diagram represents.It concludes all the reachable markings and the next state/marking of each reachable one explaining the double size of the decision diagram.

Chapter 3

Verification of the systems

Introduction The field of *verification*; the evaluation of a product, a system or a service whether they meet the required specifications for their manufacturing purposes. The verification of the systems has met great acceptance and it is widely used, after performance errors that occurred on systems which were already in the market, were discovered. The drawback of an error found after the sell of the product is that costs at each manufacturing company millions of dollars, an amount of money that no firm would like to mislay. A very famous error example was the floating point unit on the Intel P5 Pentium that was manufactured in 1994 [13]. The unit could calculate in the wrong way 1 out of 9 billion actions, and the error was obvious only after the 4 decimal numbers. Nevertheless, the 4 digital numbers that for the simple users are of no significant importance, for bankers or researchers that had to deal with the accuracy of their results (pharmacists, mathematicians) this kind of error is of great importance. For the history, the error was found by the Professor Nicely [17].

3.1 Formal Verification

Introduction *Formal Verification* [15] is the procedure of proving whether a system is correct or not, through mathematical methods. The correctness stands for the approving of a system that does meet the specifications and functionality that was built for. Formal verification is " preferred" to simulations, considering that a mathematical proof provides certainty on the evaluation of a system.This leads to the avoidance of any possible human omission that could affect the assessment of a system. There are three formal verification techniques : *Equivalence checking*, *Model checking*, *Theorem proving*.

3.1.1 Formal Verification Techniques

Equivalence checking [15] is used to assert whether two similar systems are equal or not. It is the most appropriate technique for *combinational circuits*. The similarity of the circuits is evaluated through boolean expressions, directing to the comparison of the BDDs with interpreting the circuits. The second technique is *Model checking* [15] which is used to analyze if a system meets certain properties and specifications. This method is more suitable for transition systems and automata. And the last technique *Theorem Proving* [15] asserts the definiteness of systems specifications through a theorem's proof.

The drawback of this method, though it is the most powerful, is that it needs very high educated human resources.
We focus on the examination of the behavior of a Petri Net, a transition system, with the use of the most efficient for our purpose technique, *Model checking*.

3.2 Model Checking - Verification Technique of GreatSPN

For the reasons that we have already expressed GreatSPN uses the method of model checking to evaluate the properties that the each system meets too. Two logics are used to express the properties for evaluation:*Linear Temporal Logic* and *Computational Tree Logic*.

3.2.1 Linear Temporal Logic (LTL)

Linear Temporal Logic [21] is a logic time that dependents on the events of a system. LTL encodes atomic formulas in order to determine if a possible event with eventually happen in the future or eventually will not happen. Provided that the formula which is to be checked is ϕ the operators of the linear temporal logic are

- $\mathbf{X}\phi$, the formula will be true in the next time slot
- $\psi\mathbf{U}\phi$, the formula ϕ will be definitely true in the future and surely before ψ

Any other formula with different notation in LTL can be expressed using the main operators of the logic. Two additional operators that occur of the main ones are

- $\mathbf{F}\phi$, the formula will be true some time slot in the future,Fϕ = [TRUE U ϕ]
- $\mathbf{G}\phi$, the formula will be true globally, in every single time slot, Gϕ = F ϕ

The Linear Temporal Logic is also compatible with the use of all logical operators : $\wedge$(and), $\vee$(or), and $\neg$(not). The encoding of the above LTL operators is shown in figure below.

3.2.2 Computational Tree Logic (CTL)

Computational Tree Logic [24] is also a temporal logic, used to express the properties for non deterministic system behavior. The logic's structure is tree-like and it is used to check the specifications as far as the flow information of the system described is concerned. The tree-like structure determines the different path in the future which may or may not be verified. Under this characteristics CTL is claimed to be the most suitable expressive language for model checkers. The systems that model checkers study can have more than one behavior for each time slot, thus the tree-like structure provides more information for the property than the linear logic may provide. To form an expression in CTL also the operators of LTL are required.
The main operators of CTL are:

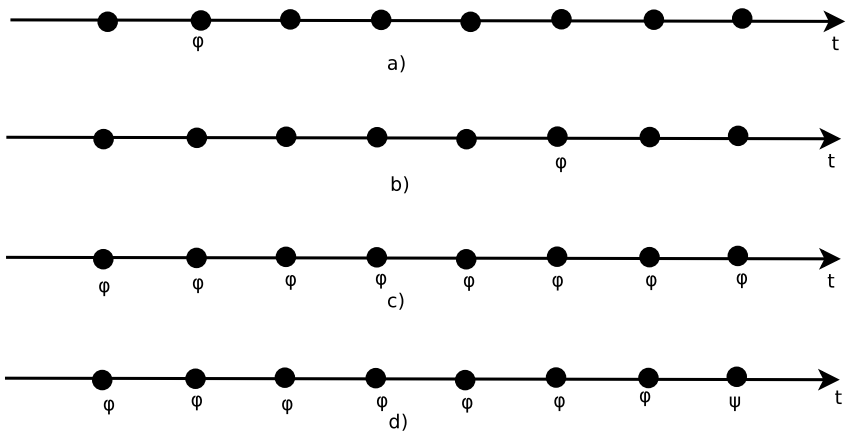

Figure 3.1: a)Xϕ , b)Fϕ, c)Gϕ, d)ψUϕ

- **A** (Xϕ, Fϕ, Gϕ, ψUϕ), the formula must be true along **all** ($\forall$) paths on the tree structure
- **E** (Xϕ, Fϕ, Gϕ, ψUϕ), the formula must be true in **at least** ($\exists$) one path on the tree structure

The formulas interpretations are shown on figure *Figure 3.2 and 3.3*. The red nodes on the images represent the formula ϕ and the green ones the formula ψ.

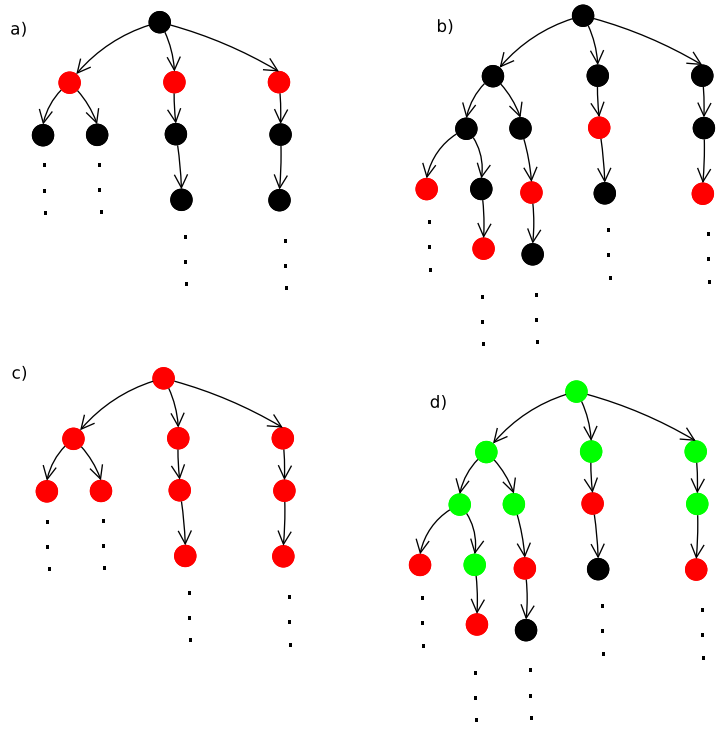

Figure 3.2: a)AXϕ , b)AFϕ, c)AGϕ, d)A(ψUϕ)

In GreatSPN tool systems are described via Petri Nets, i.e. transition system. The most suitable formal verification method to evaluate properties encoded in Petri Nets is model checking and specifically through the CTL expressions.

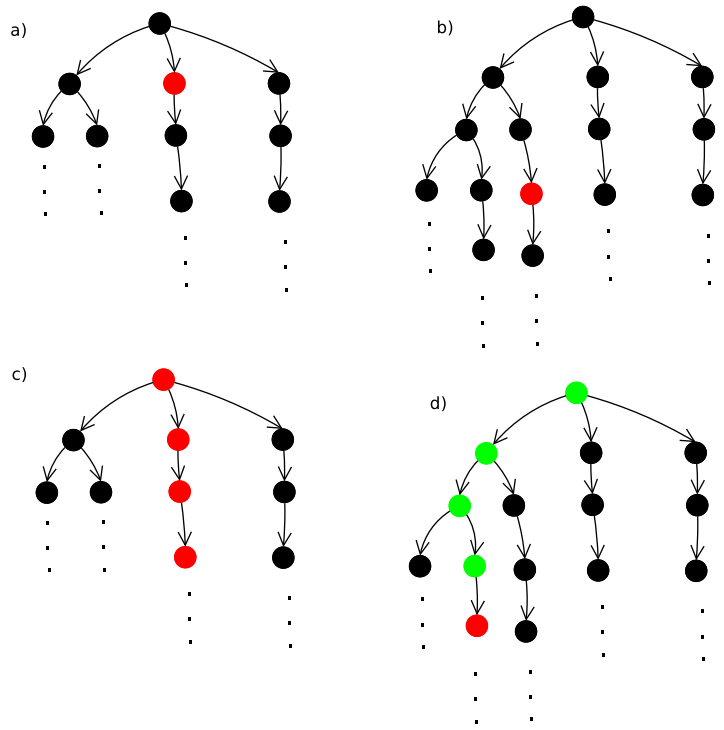

Figure 3.3: a)EXϕ , b)EFϕ, c)EGϕ, d)E(ψUϕ)

Chapter 4

Exploiting P-invariants

4.1 Checking the competitiveness of the tool between the old and the new version of GreatSPN

Our first target was to check the competitiveness of GreatSPN between the old and the new version of the tool. The last update was done before February 2012 and its correctness was not checked. To do so, we used three proposed Petri Net systems of the MCC contest [1] and three proposed properties, to check the systems' verification and simultaneously the tool's reliability. The three Petri Net systems we implemented were 1)The Kanban Model 2)The Flexible Manufacturing Model (FMS) 3)The MAPK model and the properties to be evaluated were respectively 1)The non safety of the system 2)The existence of Deadlock 3)The reversibility of the system

4.1.1 The comparison of the two different versions of the tool

The Kanban Model models a programming system that helps at deciding what product should be manufactured, when and in what quantity should be produced. *The Flexible Manufacturing System* is a fault tolerant system, that allows the system to react in case of desirable and undesirable changes that may occur. And the last model *MAPK* describes a biochemical reaction, Mitogen-Activated Protein Kinage. The graphical representation of the model is described on the following three figures.

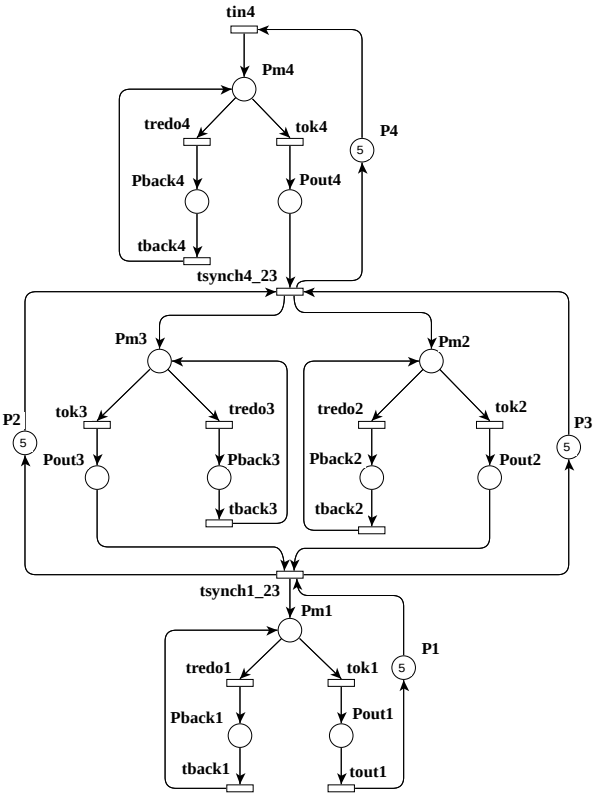

Figure 4.1: The Kanban model

The description of the behavioral properties As it is already mentioned, the properties that their correctness should be checked are expressed in the computational tree logic as it is shown below

- Non safety property: $\forall p \in P, AGm(p) > 1$, i.e. all places cannot have more than one token in each reachable marking
- Deadlock property: $AFdeadlock$,i.e. if there is a subset of places where once they are unmarked they will always be unmarked
- Reversibility property : $AGEF(M_0)$, i.e. if from each reachable marking the initial marking is recovered

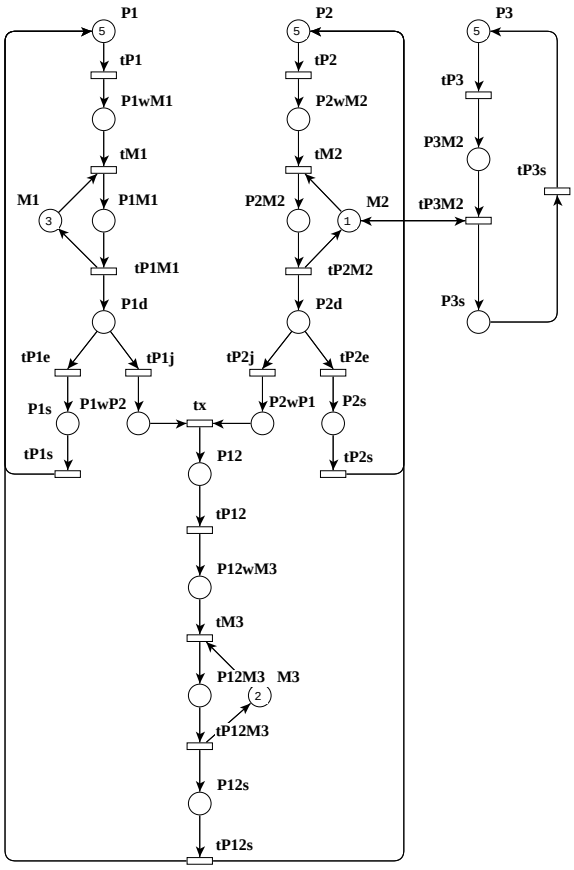

Figure 4.2: The Flexible Manufacturing System

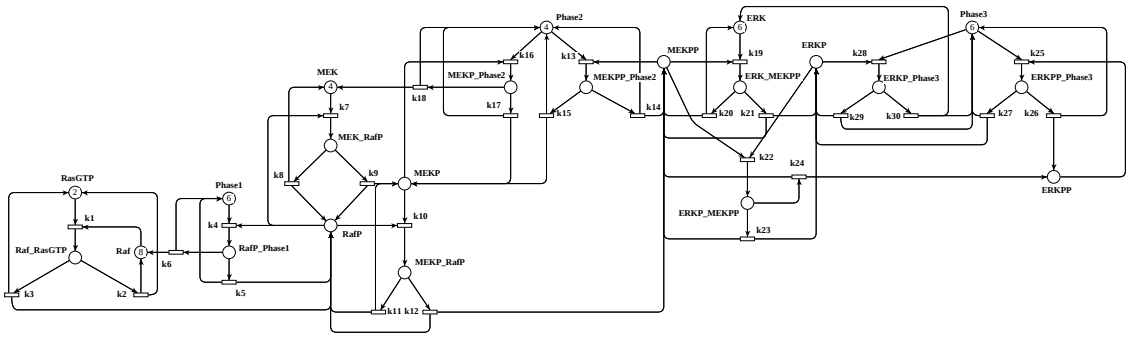

Figure 4.3: The MAPK model

4.1.2 The understanding of the results of the comparison between the two version

The results The experiments were implemented in GreatSPN and in bounded Petri Net systems. Before we proceed it is important to declare that the order of the variables is exactly the same in the both versions of the tool. GreatSPN is strongly attach to the input, since the order of the places on the MDD is the same order that the user

declares during the building of the Petri Net system via GUI. For this reason we build the Petri Net systems on the exact same order. The maximum number of tokens each time is equal to the largest multiplicity of a place on a Petri Net system.Starting from the initial marking and adding each time 5 tokens up to 30-35 token for each model, we study the behavior of the systems in both versions of the tool. We use bounded Petri Net systems to create a finite RS which leads to a finite RG. Below, the table provides in detail the discrepancy as far as time execution, time for the RS built,evaluation time for each property, the number of nodes on the MDD and the memory footprint. In the proceeding tables the variable *Bound*, declares the maximum number of tokens in each reachable marking, *RS Time Gen*, declares the timed needed for the generation of the reachability set and it is counted in seconds, as the *Eval. Time* which is the demanding time for the evaluation of the CTL property. *Pro. Eval.* is the proprty's evaluation, and it can only be {TRUE, FALSE}, *#Nodes* is the number of nodes of the created MDD, *Mem. Use* and *Mem. All.* is the memory used for each execution of the tool and the memory allocation during each experiment, respectively. Both are calculated in bytes.

The Kanban model The initial marking for the Kanban model is : $M_0 = \{5P_1 + 5P_2 + 5P_3 + 5P_4\}$. The tables that follow depict the numeric results between the two versions of GreatSPN.

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0	2.55E+006	TRUE	0.01	1,460	4.08E+004	9.71E+004
10	0.01	1.01E+009	TRUE	0.03	3,705	1.30E+005	2.59E+005
15	0.04	4.70E+010	TRUE	0.04	7,000	2.97E+005	4.59E+005
20	0.09	8.05E+011	TRUE	0.1	11,353	5.67E+005	8.95E+005
25	0.17	7.68E+012	TRUE	0.16	16,740	9.59E+005	1.43E+006
30	0.27	4.99E+013	TRUE	0.23	23,193	1.50E+006	2.15E+006
35	0.43	2.46E+014	TRUE	0.33	30,680	2.21E+006	3.05E+006

Table 4.1: Non safety Property for Kanban Model- New version of the tool

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0.01	2.55E+006	TRUE	0.01	1.476	4.14E+004	9.71E+004
10	0.01	1.01E+009	TRUE	0.02	3.721	1.31E+005	2.59E+005
15	0.04	4.70E+010	TRUE	0.07	7.016	2.98E+005	4.59E+005
20	0.08	8.05E+011	TRUE	0.1	11.361	5.67E+005	8.95E+005
25	0.16	7.68E+012	TRUE	0.19	16.756	9.61E+005	1.43E+006
30	0.28	4.99E+013	TRUE	0.31	23.201	1.50E+006	2.15E+006
35	0.43	2.46E+014	TRUE	0.4	30.696	2.22E+006	3.05E+006

Table 4.2: Non safety Property for Kanban Model- Old version of the tool

For the Kanban model and regarding the non safety property, we can mark that though the time for RS generation is better in the old version of the tool, in the new one it appears that we gain better time in the evaluation time of the property. Still the memory allocation, the memory used and the number of nodes on the MDD is exactly the same in both version of GreatSPN. In both versions the maximum number of tokens is 35.

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0.01	2.55E+006	FALSE	0.01	471	1.31E+004	7.96E+004
10	0.01	1.01E+009	FALSE	0.01	1,588	5.50E+004	1.32E+005
15	0.04	4.70E+010	FALSE	0.04	3,264	1.37E+005	2.37E+005
20	0.08	8.05E+011	FALSE	0.09	5,397	2.66E+005	4.59E+005
25	0.17	7.68E+012	FALSE	0.12	6,888	3.83E+005	6.22E+005
30	0.27	4.99E+013	FALSE	0.22	11,887	7.66E+005	1.31E+006
35	0.44	2.46E+014	FALSE	0.35	14,108	9.93E+005	1.35E+006

Table 4.3: Deadlock Property for Kanban Model- New version of the tool

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0	2.55E+006	FALSE	0.01	565	1.56E+004	8.17E+004
10	0.02	1.01E+009	FALSE	0.02	1.685	5.89E+004	1.42E+005
15	0.04	4.70E+010	FALSE	0.05	3.405	1.44E+005	2.37E+005
20	0.08	8.05E+011	FALSE	0.08	5.725	2.85E+005	4.59E+005
25	0.16	7.68E+012	FALSE	0.15	8.645	4.95E+005	7.48E+005
30	0.27	4.99E+013	FALSE	0.21	12.165	7.88E+005	1.31E+006
35	0.44	2.46E+014	FALSE	0.33	16.285	1.18E+006	1.82E+006

Table 4.4: Deadlock Property for Kanban Model- Old version of the tool

Observing the results for the deadlock property, it is apparent that there is no significant difference neither in time nor or memory footprint. The two different versions of the tool have the same behavior for the Kanban Model and the deadlock property. In both versions the maximum number of tokens studied are 35.

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0.01	2.55E+006	TRUE	0.2	18,039	5.11E+005	9.12E+005
10	0.01	1.01E+009	TRUE	2.33	107,863	3.87E+006	6.75E+006
15	0.04	4.70E+010	TRUE	9.14	385,143	1.67E+007	2.65E+007
20	0.07	8.05E+011	TRUE	30.45	807,172	4.03E+007	6.27E+007
25	0.17	7.68E+012	TRUE	70.09	1,611,212	9.31E+007	1.34E+008

Table 4.5: Reversibility Property for Kanban Model- New version of the tool

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0.01	2.55E+006	TRUE	0.2	22.846	6.45E+005	1.18E+006
10	0.01	1.01E+009	TRUE	3.98	223.798	7.90E+006	1.28E+007
15	0.03	4.70E+010	TRUE	76.51	1.815.202	6.80E+007	1.41E+008

Table 4.6: Reversibility Property for Kanban Model- Old version of the tool

In this property it is clear that the new version of the tool is more powerful since it can perform the evaluation of the property in the Kanban model for a bound of 25 tokens, instead of 15 tokens that can perform the old version of the tool. For 20 maximum tokens the old version of the tool crashes.

The Flexible Manufacturing Model (FMS) The initial marking for the FMS model is : $M_0 = \{5P_1 + 5P_2 + 5P_3 + 3M_1 + 1M_2\}$ the behavior of the tool is shown on the following tables

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0.07	2.90E+006	TRUE	0.1	19,588	5.26E+005	1.00E+006
10	0.82	2.50E+009	TRUE	1.1	132,138	4.20E+006	6.67E+006
15	3.45	2.17E+011	TRUE	3.78	414,229	1.51E+007	2.50E+007
20	9.91	6.03E+012	TRUE	10.13	944,965	3.90E+007	5.81E+007
25	23.36	8.54E+013	TRUE	21.61	1,803,184	8.28E+007	1.30E+008
30	49.05	7.74E+014	TRUE	40.36	3,067,639	1.55E+008	2.38E+008
35	86.84	5.09E+015	TRUE	77.75	4,817,109	2.67E+008	3.78E+008

Table 4.7: Non Safety Property for FMS Model- New version of the tool

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0.07	2.90E+006	TRUE	0.1	19.729	5.34E+005	9.92E+005
10	0.67	2.50E+009	TRUE	0.87	128.683	4.12E+06	6.53E+006
15	2.83	2.17E+011	TRUE	3.46	404.163	1.48E+07	2.46E+007
20	8.19	6.03E+012	TRUE	9.04	925.343	3.84E+07	5.72E+007
25	19.22	8.54E+013	TRUE	18.49	1.771.023	8.17E+07	1.29E+008
30	38.56	7.74E+014	TRUE	36.54	3.019.953	1.54E+08	2.36E+008
35	0.46	5.09E+015	TRUE	75.62	4.750.883	2.64E+08	3.74E+008

Table 4.8: Non Safety Property for FMS Model- Old version of the tool

Observing the results we notice that the old version of the tool offers better time results in the generation of the RS and in memory use.

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0.08	2.90E+006	FALSE	0.09	10.808	2.93E+005	5.19E+005
10	0.8	2.50E+009	FALSE	0.89	64.367	2.06E+006	3.69E+006
15	3.45	2.17E+011	FALSE	3.56	203.271	7.53E+006	1.33E+007
20	9.91	6.03E+012	FALSE	9.53	437.929	1.83E+007	2.85E+007
25	23.87	8.54E+013	FALSE	21.55	817.226	3.87E+007	6.08E+007
30	49.28	7.74E+014	FALSE	42.22	1.604.184	8.19E+007	1.29E+008
35	87.06	5.09E+015	FALSE	70.91	2.414.959	1.35E+008	2.12E+008

Table 4.9: Deadlock Property for FMS Model- New version of the tool

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0.09	2.90E+006	FALSE	0.09	11.597	3.13E+005	5.31E+005
10	0.74	2.50E+009	FALSE	0.69	74.807	2.38E+006	4.06E+006
15	3.05	2.17E+011	FALSE	2.63	234.692	8.57E+006	1.39E+007
20	8.6	6.03E+012	FALSE	7.42	536.252	2.21E+007	3.46E+007
25	21.91	8.54E+013	FALSE	17.45	1.024.487	4.69E+007	7.33E+007
30	45.94	7.74E+014	FALSE	33.77	1.744.397	8.79E+007	1.33E+008
35	82.62	5.09E+015	FALSE	63.95	2.268.408	1.23E+008	1.88E+008

Table 4.10: Deadlock Property for FMS Model- New version of the tool

For the deadlock property on the same model the older version seems to have a better reaction as time is concerned, either for the time needed for the RS generation or for

the time needed for the property's evaluation. The new version of the tool has a better behavior in memory used and memory allocation for the property's performance.

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0.08	2.90E+006	FALSE	0.11	14,629	3.96E+005	7.24E+005
10	0.79	2.50E+009	FALSE	1.11	89,254	2.86E+006	5.11E+006
15	3.55	2.17E+011	FALSE	4.08	245,989	9.17E+006	1.58E+007
20	9.99	6.03E+012	FALSE	11.05	581,844	2.44E+007	3.77E+007
25	24.04	8.54E+013	FALSE	24.5	1,082,733	5.08E+007	8.12E+007
30	50.51	7.74E+014	FALSE	50.12	1,765,746	9.22E+007	1.48E+008
35	87.57	5.09E+015	FALSE	89.81	2,704,339	1.55E+008	2.47E+008

Table 4.11: Reversibility Property for FMS Model- New version of the tool

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
5	0.08	2.90E+006	FALSE	0.49	46.034	1.29E+006	2.35E+006
10	0.72	2.50E+009	FALSE	58.49	2.301.029	7.11E+007	1.38E+008

Table 4.12: Reversibility Property for FMS Model- Old version of the tool

In the reversibility property the new version has a better execution since it performs for all possible initial markings instead of the old version of the tool that crashes after the initial marking has a bound of 10 tokens.

Mitogen Activated Protein Kinage model (MAPK) The initial marking for the MAPK model is : $M_0 = \{2RaGTP + 4MEK + 6Phase1 + 4Phase2 + 6Phase3 + 4MEK + 6ERK + 8Raf\}$ .The behavior of the tool for the three properties is shown on the following tables

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
8	0.21	1.93E+010	TRUE	0.01	3.622	1.34E+005	3.17E+005
13	0.99	3.98E+013	TRUE	0.05	11.835	5.54E+005	1.14E+006
18	2.52	6.71E+015	TRUE	0.11	29.809	1.69E+006	2.88E+006
23	5.91	3.36E+017	TRUE	0.21	53.49	3.62E+006	7.03E+006
28	10.98	7.99E+018	TRUE	0.36	114.298	8.87E+006	1.41E+007
33	20.88	1.15E+020	TRUE	0.55	169.238	1.46E+007	2.25E+007
38	35.85	1.15E+021	TRUE	0.83	274.762	2.68E+007	3.87E+007

Table 4.13: Non Safety Property for MAPK Model- New version of the tool

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
8	0.08	1.93E+010	TRUE	0.03	3.552	1.31E+005	3.01E+005
13	0.46	3.98E+013	TRUE	0.06	13.282	6.24E+005	1.09E+006
18	1.72	6.71E+015	TRUE	0.14	32.962	1.88E+006	3.14E+006
23	4.06	3.36E+017	TRUE	0.28	66.092	4.42E+006	7.11E+006
28	9.11	7.99E+018	TRUE	0.57	116.172	8.92E+006	1.41E+007
33	20.29	1.15E+020	TRUE	0.95	186.712	1.62E+007	2.25E+007
38	32.25	1.15E+021	TRUE	0.89	281.192	2.72E+007	4.34E+007

Table 4.14: Non Safety Property for MAPK Model- Old version of the tool

In this model the most observing differences appear on the time generation of the RS, which is better in the old version and on the time of the property's evaluation, that is better on the new version.

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
8	0.23	1.93E+010	FALSE	0.01	3.622	1.34E+005	2.95E+005
13	0.98	3.98E+013	FALSE	0.07	11.835	5.54E+005	1.10E+006
18	2.56	6.71E+015	FALSE	0.17	29.809	1.69E+006	2.84E+006
23	5.81	3.36E+017	FALSE	0.34	53.491	3.62E+006	6.81E+006
28	10.84	7.99E+018	FALSE	0.59	114.298	8.87E+006	1.39E+007
33	20.84	1.15E+020	FALSE	1.01	169.238	1.46E+007	2.18E+007
38	35.04	1.15E+021	FALSE	1.5	274.762	2.68E+007	3.77E+007

Table 4.15: Deadlock Property for MAPK Model- New version of the tool

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
8	0.09	1.93E+010	FALSE	0.03	3.552	1.31E+005	2.80E+005
13	0.45	3.98E+013	FALSE	0.07	13.282	6.24E+005	1.05E+006
18	1.76	6.71E+015	FALSE	0.21	32.962	1.88E+006	3.11E+006
23	4.62	3.36E+017	FALSE	0.31	66.092	4.42E+006	6.95E+006
28	9.13	7.99E+018	FALSE	0.78	116.172	8.92E+006	1.39E+007
33	19.99	1.15E+020	FALSE	0.91	186.712	1.62E+007	2.18E+007
38	37.4	1.15E+021	FALSE	1.75	281.192	2.72E+007	4.24E+007

Table 4.16: Deadlock Property for MAPK Model- Old version of the tool

The results in this case are mixed, in the sense that for an initial marking the time for the evaluation of the property might be better in the old version, and for another might

be better in the new version. The same is shown also for the memory mapping and the memory allocation. The only different consistency is about the time generation of the RS which much better in the older version of the tool.

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
8	0.23	1.93E+010	FALSE	10.59	351.199	1.41E+007	2.37E+007

Table 4.17: Reversibility Property for MAPK Model- New version of the tool

Bound	RS Time Gen.	RS Size	Prop.Eval.	Eval.Time	#Nodes	Mem.Use	Mem.All.
8	0.08	1.93E+010	FALSE	10.84	506.916	2.03E+007	3.29E+007
13	0.44	3.98E+013	FALSE	24.35	852.443	4.28E+007	6.19E+007

Table 4.18: Reversibility Property for MAPK Model- Old version of the tool

In the reversibility property the problem is that the tool crashes too quickly and we can not have a clear idea for which version the results are better either on time calculation or on memory footprint.

Driven by the inconsistency of the behavior of the tool, since for all the above Petri Net systems the behavior was not stable for either one of the version, we decided to focus on making the tool more efficient. Our first aim is to try to minimize the number of nodes of the MDD used to describe the RS, which will lead to smaller memory footprint, which might lead to better time results. Additionally, since the variable ordering on the MDD is strongly connected to the user, we will aim at doing the tool user-input independent.As stated before the problem of the optimal ordering of an MDD is an NP-complete problem [7], thus we will try to approach the optimal solution proposing an heuristic algorithm.

4.2 Reordering of the Places

Based on [12], [10], [23] we decided to focus on the information provided by the functional dependencies of the Petri Net systems to built smaller MDDs. Since merging the nodes of an MDD guarantees smaller number of nodes, rather than eliminating them [10], we will experiment on three small Petri Net systems to detect the impact that the variable ordering has to the number of nodes of the decision diagrams. Precisely, the center of our attraction will be on the P-invariants, since for the symbolic generation of the RS exploiting the P-invariants offers better results [7], in sense that we will gain a decision diagram with less nodes. Depending on P-invariants in advance, we gain *a priori* relationships that hold for all reachable markings. It is important for exploiting the P-invariants, that all places are covered; i.e. all places must be part of at least one invariant,

otherwise we are not able to be aware of the total information needed. Experiments were performed on the GreatSPN tool.

4.2.1 Reordering the places of the three model Petri Net Systems

First Petri Net System The first model we implemented ,consists of three independent Petri Nets, that basically depicts three independent users.The results that are shown in the table below, indicate that the order of the places according to the P-invariants are more efficient for encoding the RS on MDD. The P-invariants obtained are:

P-invariant1 : P1, PQ, P2
P-invariant2 : S1, SQ, S2
P-invariant3 : R1, RQ, R2

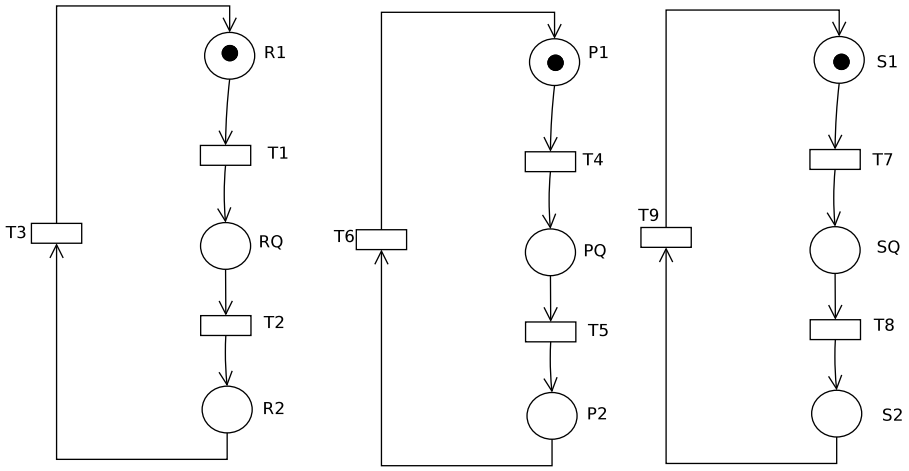

Figure 4.4: Petri Net consisted of three independent Petri Nets

The results after reordering the places are shown in the table below:

Order of var	RS Size	Peak Nodes	Final Nodes	Peak Mem Used
R1, RQ, R2, P1, PQ, P2 , S1, SQ, S2	27	25	23	552
R1, P1, S1, RQ, PQ, SQ, R2, P2, S2	27	57	51	1.268
R1, R2, P1, P2, S1, S2, RQ, PQ, SQ	27	61	43	1.352
P1, PQ, P2, S1, SQ, S2, R1, RQ, R2	27	25	23	552
S1, SQ, S2, P1, PQ, P2, R1, RQ, R2	27	25	23	552
R1, RQ, R2, S1, SQ, S2, P1, PQ, P2	27	25	23	552

Table 4.19: First Model (Places = 9 & Bound = 1)

The P-invariants obtained in this system are as we can see independent to each other; i.e. each place is in only one P-invariant. As we can see the results are better when the places are ordered according to the P-invariants. The last three lines of the above table demonstrate that the order of the places is important but not the order of the appearance of each invariant. For example, we gain the same results either for *P1, PQ, P2, S1, SQ, S2, R1, RQ, R2* or for *S1, SQ, S2, P1, PQ, P2, R1, RQ, R2*. It is of no importance the order that each invariants appears (P-invariant1, P-invariant2, P-invariant3 or P-invariant2,P-invariant1,P-invariant3) as long as the places that are contained in the same P-invariant are not mixed as in the case that is shown on the second and third line of the table above. In the "mixed" case we get worst results in the number of nodes and in memory footprint.

Second Petri Net System The second Petri Net that was implemented consists of 11 places, two of which are shared among three independent Petri Nets. The second Petri Net depicts three users who require for access on the same serversource. The difference is that the P-invariants in this net are not independent but there are common places in more than one P-invariant.The P-invariants obtained are :

P1 : P1, PQ, P2
P2 : SQ, Q1, Q2, PQ, RQ
P3 : S1, SQ, S2
P4 : R1, RQ, R2

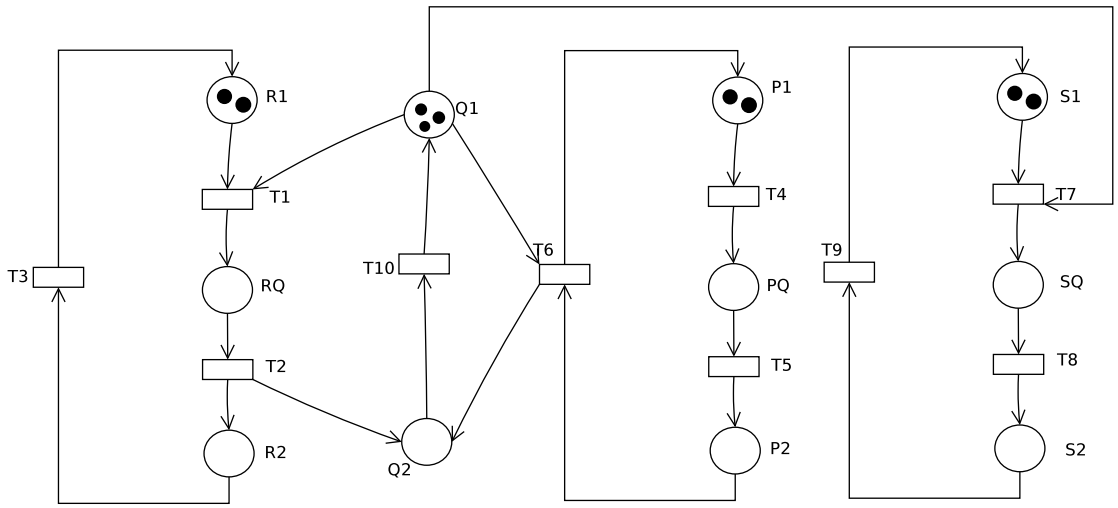

Figure 4.5: Petri Net which consists of two shared places

The obtained results after reordering the places are in the table below:

Table 4.20: Second Model (Places = 11 & Bound = 3)

Order of var	RS Size	Peak Nodes	Final Nodes	Peak Mem Used
R1, RQ, R2, Q1, Q2, P1, PQ, P2, S1, SQ, S2	440	69	62	1.664
Q1, Q2, RQ, PQ, SQ, R1, R2, P1, P2, S1, S2	440	141	109	3.348
R1, R2, RQ, Q1, Q2, PQ, P1, P2, S1, S2, SQ	440	72	57	1.748
R1, P1, S1, R2, P2, S2, RQ, PQ, SQ, Q1, Q2	440	566	150	13.260
P1, PQ, P2, S1, SQ, S2, R1, RQ, R2, Q1, Q2	440	215	78	5.092
P1, PQ, P2, Q1, Q2, R1, RQ, R2, S1, SQ, S2	440	69	62	1.664
P1, PQ, P2, R1, RQ, R2, Q1, Q2, S1, SQ, S2	440	115	62	2.752

Table 4.21: Second Model (Places = 11 & Bound = 3)

In this system the influence of the P-invariants is not as clear as in the previous one, due to the existence of the common places in P-invariants. In this system we observe that the second P-invariant obtained can create a more complex situation, since the other 3 P-invariant are in a way independent. The higher we keep the shared places, common places appear before the non common in the order of the places,{Q1,Q2} the more efficient results we take, since the MDD will not have to make ¡¡ big steps ¿¿ or more formally create more states to reach every new reachable marking. For example if we see the first and the last line of the table, the only difference is that the shared places in the last line are not ordered in the first places so in order to built the reachability set it will be necessary to create more nodes for the RS construction.

Third Petri Net The third Petri Net that we have implemented is a fork and join model [5]. Also in this net there are places that are interfering with each other, in different P-invariants.The P-invariants that are obtained are :

P1 : P1, P3, P4, P6
P2 : P1, P2, P5, P6

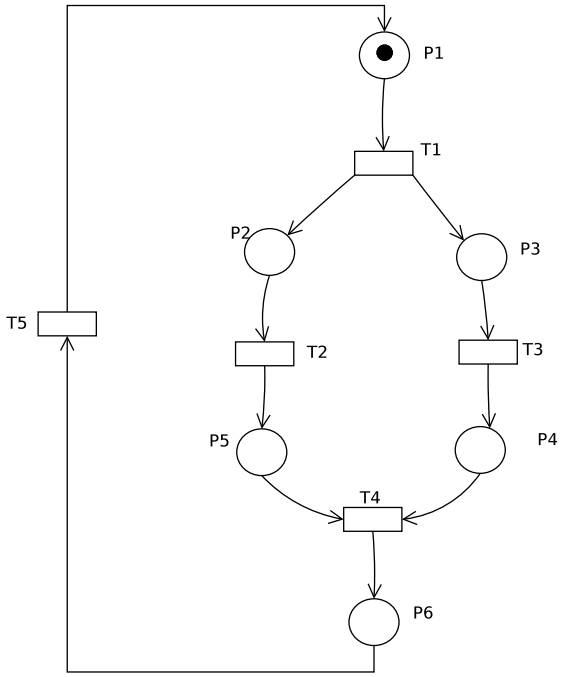

Figure 4.6: Fork and join Petri Net

Order of var	RS Size	Peak Nodes	Final Nodes	Peak Mem Used
P1, P2, P3, P4, P5, P6	196	97	95	2.728
P1, P3, P4, P2, P5, P6	196	82	79	2.140
P6, P1, P2, P3, P5, P4	196	184	81	5.452
P1, P3, P4, P6, P2, P5	196	91	64	2.480
P1, P2, P5, P3, P4, P6	196	82	79	2.140
P2, P5, P1, P6, P3, P4	196	76	49	2.100
P3, P4, P1, P6, P2, P5	196	76	49	2.100

Table 4.22: Third Model (Places = 6 & Bound =5)

In this example the influence on the output of the P-invariant is still obvious. The last two lines of the table demonstrate the best results in all different orderings provided to the fact the sharedcommon places {P1, P2} are ordered close enough and between the non common places of the P-invariants in which are concluded.

From the results provided in the former images, it is clear that the closer the common places are kept in their ordering smaller mdds are obtained.

4.3 Proposing a new heuristic algorithm

By examining the results from the three small Petri Net systems, we propose a new heuristic algorithm exploiting the P-invariant in order to gain better results in time calculation and memory footprint. The main idea of the algorithm is to keep the sharedcommon places in the P-invariants as close as possible. Starting from the maximum

intersection among the first P-invariant and the rest of the P-invariants, max(P1$\cap P_i$), we first order the non common places of the P1, after the common places and finally the rest places of the P_i. Then we continue the same procedure with the difference that for the rest iterations, needed so as to take into account all the P-invariants, we take the minimum intersection. The common places are fixed, it is not allowed to reorder them. On the contrary, the places that are found in common after their first placement (previous the were non common places), they must be reordered. The reason that we first take the maximum intersection and after the minimum is that we want to avoid the continuous moving of the places. The pseudo-code we propose is shown in Algo 1.

Hereafter, we report the data structure used in the algorithm:

- $P = \{p\}$ the set of places in the PN;
- π_i is a bag over P, which encodes a P-invariant;
- $\Pi = \{\pi_i\}$ is a set of P-invariants;
- L is a list of places. It is used to store the variable ordering. Observe that an isomorphic is defined between places and MDD variables.
- C is a set of fixed common places among the P-invariants

Moreover, the following operators are needed:

- $p\_intersection(\pi_i, \pi_j) = \{p | \pi_i(p) > 0 \wedge \pi_j(p) > 0\}$;
- $\pi\_intersection(\pi_i) = |\{\pi_j | p\_intersection(\pi_i, \pi_j) > 0\}|$;
- $\pi\_places(\pi_i) = \{p | \pi_i(p) > 0\}$;
- *insert_at_end(L,P)* takes in input L a list of place and P a set of places and inserts each place of P in L if it is not already present.
- *remove_and_reorder_places(L,P)* takes in input L a list of place and P a set of places and removes each places of P from L and reorders L by moving one place up or more depending on the places that were removed : $L[i-1] = L[i]$
- *add_places(C,P)* takes an input C a set of places and P a set of places and adds each place of P to C if it is not already present

We will examine the results obtained after the new static ordering of the variables-places on the MDD. The new algorithm is implemented in C language.We focus on the study of the FMS model, which behavior was most unstable. The tables that follow show the difference in the time calculation and memory footprint among the new version of the tool with and without the implementation of the new proposed heuristic algorithm. The experiments were implemented in the GreatSPN tool and were performed with and without CTL properties. In the columns of the preceding tables we have add the P-invariant columns which declare that the ordering is according to the new algorithm if it is "**Y**" and the order is according to the user if is "**N**". Also the results of the two Petri

Algorithm 1 A new variable ordering heuristic for efficient encoding RS on MDD

function ALGO(Π)
  $L = \emptyset$
  $C = \{\emptyset\}$
  Take $i|\pi\_intersection(\pi_i)$ is minimum over all $\pi_i \in \Pi$
  $\Pi = \Pi \setminus \pi_i$
  $curr{=}\pi_i$
  while ($\Pi \neq \emptyset$) **do**
    Take $j|p\_intersection(curr, \pi_j)$ is maximum over all $\pi_j \in \Pi$
    $insert\_at\_end(L, \pi\_places(\pi_i) \setminus C)$
    $remove\_and\_reorder\_places(L, p\_intersection(L\_\text{places}(L), \pi_j) \setminus C)$
    $add\_places(C, p\_intersection(curr, \pi_j))$
    $insert\_at\_end(L, p\_intersection(curr, \pi_j) \setminus C)$
    $insert\_at\_end(L, \pi\_places(\pi_j) \setminus C)$
    $\Pi = \Pi \setminus \pi_j$
    $curr{=}\pi_j$
  end while
end function

Net systems that were proposed by the *MCC* competition are provided, for the *FMS model* and for the *Kanban Model*. The new algorithm can not be performed for *MAPK model* since not all the places of the model are covered by the P-invariants.

The first row shows the output of the new version of the tool without the new proposed heuristic algorithm and the second row, of each pair of lines, shows the results with the implementation of the new heuristic algorithm.

Bound	P-inv	RS Time Gen.	RS Size	#Nodes	Mem.Use	Mem.All.
5	N	0.01	2.90E+06	3.184	8.56E+004	1.88E+005
5	Y	0.02	2.90E+06	861	2.20E+004	1.05E+005
10	N	1.0	2.50E+09	19.059	6.02E+005	1.08E+006
10	Y	0.07	2.50E+09	3.187	9.59E+004	2.07E+005
15	N	4.0	2.17E+11	58.182	2.10E+006	3.17E+006
15	Y	0.32	2.17E+11	7.731	2.67E+005	4.59E+005
20	N	10.0	6.03E+12	131.060	5.34E+006	8.49E+006
20	Y	0.74	6.03E+12	14.494	5.68E+005	9.82E+005
25	N	25.0	8.54E+13	248.184	1.12E+007	1.59E+007
25	Y	1.72	8.54E+13	28.342	1.27E+006	2.29E+006
30	N	52.0	7.74E+14	420.060	2.09E+007	3.05E+007
30	Y	3.12	7.74E+14	44.140	2.19E+006	3.41E+006
35	N	91.0	5.09E+15	657.185	3.57E+007	5.21E+007
35	Y	5.54	5.09E+15	63.165	3.43E+006	5.65E+006

Table 4.23: FMS model without the evaluation of a CTL property- New version of the tool without and with the new heuristic algorithm

The first property for evaluation is $A\ F\ deadlock$, if starting from the initial marking in all paths, some time in the future there is a deadlock. There is a subset of places where if once they are empty (are without tokens) there will be always empty. The evaluation of the property is FALSE, e.i. no deadlock is met in the FSM.

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.08	2.90E+006	FALSE	0.09	10.808	2.93E+005	5.19E+005
5	Y	0.02	2.90E+006	FALSE	0.01	1.468	3.81E+004	1.20E+005
10	N	0.84	2.50E+009	FALSE	0.92	64.367	2.06E+006	3.69E+006
10	Y	0.09	2.50E+009	FALSE	0.03	4.348	1.33E+005	2.90E+005
15	N	4.14	2.17E+011	FALSE	3.8	203.271	7.53E+006	1.33E+007
15	Y	0.32	2.17E+011	FALSE	0.07	9.301	3.27E+005	7.07E+005
20	N	11.55	6.03E+012	FALSE	9.85	434.929	1.83E+007	2.85E+007
20	Y	0.71	6.03E+012	FALSE	0.14	15.798	6.38E+005	1.32E+006
25	N	27.35	8.54E+013	FALSE	21.38	817.226	3.87E+007	6.08E+007
25	Y	1.62	8.54E+013	FALSE	0.22	28.342	1.27E+006	2.58E+006
30	N	57.82	7.74E+014	FALSE	43.54	1.604.184	8.19E+007	1.29E+008
30	Y	2.98	7.74E+014	FALSE	0.4	44.140	2.19E+006	3.80E+006
35	N	101.13	5.09E+015	FALSE	70.64	2.414.959	1.35E+008	2.12E+008
35	Y	5.3	5.09E+015	FALSE	0.52	63.165	3.43E+006	6.27E+006

Table 4.24: FMS model evaluating the property- AF deadlock- New version of the tool without and with the new heuristic algorithm

The second property for verification is AG(M1=0→(P1wM1)<N), if in all paths from the initial marking, exists globally that once the place M1=0, the machine 1 stops working (is with no tokens), implies that the place P1wM1 has less products than the upper bound of the system, i.e. the upper bound declares the products asked to be produced.

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.09	2.90E+006	TRUE	0.02	5.588	1.53E+005	3.08E+005
5	Y	0.02	2.90E+006	TRUE	0.01	861	2.20E+004	1.05E+005
10	N	0.89	2.50E+009	TRUE	0.15	35.302	1.14E+006	1.90E+006
10	Y	0.07	2.50E+009	TRUE	0.01	3.187	9.59E+004	2.07E+005
15	N	4.07	2.17E+011	TRUE	0.62	108.617	4.05E+006	6.76E+006
15	Y	0.01	2.17E+011	TRUE	0.01	7.731	2.67E+005	4.59E+005
20	N	11.42	6.03E+012	TRUE	1.58	245.032	1.04E+007	1.60E+007
20	Y	0.73	6.03E+012	TRUE	0.01	14.494	5.68E+005	9.82E+005
25	N	29.75	8.54E+013	TRUE	3.85	464.047	2.19E+007	3.54E+007
25	Y	1.6	8.54E+013	TRUE	0.02	28.342	1.27E+006	2.29E+006
30	N	55.97	7.74E+014	TRUE	6.41	785.162	4.10E+007	6.53E+007
30	Y	2.99	7.74E+014	TRUE	0.02	44.140	2.19E+006	3.41E+006
35	N	99.17	5.09E+015	TRUE	10.85	1.227.877	7.03E+007	1.03E+008
35	Y	5.27	5.09E+015	TRUE	0.03	63.165	3.43E+006	5.65E+006

Table 4.25: FMS model evaluating the property- AG(M1=0→(P1wM1)<N) - New version of the tool without and with the new heuristic algorithm

The third property is AG(M2=0→(P3M2+P2wM2)<7), if in all paths starting from the initial marking, exists globally, that when M2=0,the machine number 2 is empty (stops working), the sum of the products that are produced in the places P3M2 + P2wM2 is less that 7. Which is FALSE since the maximum number of the products starts from 5 and is increased.

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.1	2.90E+006	FALSE	1.1	79.686	2.20E+006	3.93E+006
5	Y	0.02	2.90E+006	FALSE	0.48	49.679	1.29E+006	2.37E+006
10	N	0.92	2.50E+009	FALSE	8.43	434.114	1.40E+007	2.29E+007
10	Y	0.13	2.50E+009	FALSE	3.54	206.675	6.39E+006	1.09E+007
15	N	4.03	2.17E+011	FALSE	36.88	1.471.135	5.42E+007	8.84E+007
15	Y	0.32	2.17E+011	FALSE	9.6	473.861	1.68E+007	2.88E+007
20	N	11.24	6.03E+012	FALSE	101.85	3.183.822	1.07E+008	2.10E+008
20	Y	0.93	6.03E+012	FALSE	19.02	845.651	3.38E+007	5.67E+007
25	N	X	X	X	X	X	X	X
25	Y	1.94	8.54E+013	FALSE	32.13	1.430.459	6.38E+007	9.51E+007
30	N	X	X	X	X	X	X	X
30	Y	3.35	7.74E+014	FALSE	46.85	1.951.288	9.48E+007	1.47E+008
35	N	X	X	X	X	X	X	X
35	Y	6.33	5.09E+015	FALSE	66.33	2.771.764	1.48E+008	2.17E+008

Table 4.26: FMS model evaluating the property- AG(M2=0→(P3M2+P2wM2)<7) - New version of the tool without and with the new heuristic algorithm

The next property is EF(M1!=0 and not en(tM1)), and checks if there exists at least one path where some time in the future while the machine 1 (M1) is not empty (is working) and the transition tM1, in which the place M1 is the input place, isn't enabled. The property is TRUE.

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.09	2.90E+006	TRUE	0.48	47.029	1.34E+006	2.41E+006
5	Y	0.02	2.90E+006	TRUE	0.03	5.045	1.37E+005	2.54E+005
10	N	0.85	2.50E+009	TRUE	9.6	520.605	1.81E+007	3.05E+007
10	Y	0.09	2.50E+009	TRUE	0.04	6.330	1.93E+005	3.71E+005
15	N	4.21	2.17E+011	TRUE	6.29	362.078	1.37E+007	2.20E+007
15	Y	0.43	2.17E+011	TRUE	1.04	64.712	2.65E+006	4.43E+006
20	N	X	X	X	X	X	X	X
20	Y	0.83	6.03E+012	TRUE	0.23	25.993	1.04E+006	1.64E+006
25	N	X	X	X	X	X	X	X
25	Y	1.84	8.54E+013	TRUE	5.66	257.104	1.39E+007	2.36E+007
30	N	X	X	X	X	X	X	X
30	Y	3.74	7.74E+014	TRUE	9.5	454.870	2.86E+007	4.36E+007
35	N	X	X	X	X	X	X	X
35	Y	6.62	5.09E+015	TRUE	21.15	771.689	5.13E+007	7.42E+007

Table 4.27: FMS model evaluating the property- EF(M1!=0 and not en(tM1)) - New version of the tool without and with the new heuristic algorithm

The property number 5 is EG not (P1s=N and P2s=N and P3s=N), if there is at least one path where globally the number of products in the places P1s, P2s and P3s is not equivalent to the maximum number of them. This property is evaluated as TRUE.

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.1	2.90E+006	TRUE	0.09	11.151	3.01E+005	5.38E+005
5	Y	0.02	2.90E+006	TRUE	0.01	1.758	4.58E+004	1.32E+005
10	N	0.93	2.50E+009	TRUE	0.94	65.323	2.09E+006	3.70E+006
10	Y	0.09	2.50E+009	TRUE	0.03	4.720	1.45E+005	3.00E+005
15	N	4.06	2.17E+011	TRUE	3.8	205.265	7.59E+006	1.35E+007
15	Y	0.36	2.17E+011	TRUE	0.12	10.156	3.61E+005	7.07E+005
20	N	11.72	6.03E+012	TRUE	9.73	441.287	1.84E+007	2.85E+007
20	Y	0.93	6.03E+012	TRUE	0.18	16.613	6.63E+005	1.33E+006
25	N	27.38	8.54E+013	TRUE	21.8	822.723	3.89E+007	6.14E+007
25	Y	2.3	8.54E+013	TRUE	0.39	28.342	1.27E+006	2.58E+006
30	N	56.08	7.74E+014	TRUE	41.89	1.612.401	8.22E+007	1.29E+008
30	Y	3.86	7.74E+014	TRUE	0.67	44.140	2.19E+006	3.83E+006
35	N	99.48	5.09E+015	TRUE	70.3	2.423.899	1.35E+008	2.12E+008
35	Y	6.86	5.09E+015	TRUE	0.8	63.156	3.43E+006	6.27E+006

Table 4.28: FMS model evaluating the property- EG not (P1s=N and P2s=N and P3s=N) - New version of the tool without and with the new heuristic algorithm

And the last property for evaluation is E[(M1>0)U(P1s=N and P2s=N and P3s=N)], if exists at least one path from the initial marking that the machine 1 (M1) works until the products in the places P1s, P2s, P3s are created. The property is TRUE.

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.08	2.90E+006	TRUE	4.17	278.913	7.38E+006	1.27E+007
5	Y	0.01	2.90E+006	TRUE	0.76	64.163	1.69E+006	3.08E+006
10	N	0.87	2.50E+009	TRUE	96.59	4.155.299	1.11E+008	2.28E+008
10	Y	0.12	2.50E+009	TRUE	9.5	502.213	1.53E+007	2.66E+007
15	N	X	X	X	X	X	X	X
15	Y	0.42	2.17E+011	TRUE	43.61	1.899.377	6.48E+007	1.06E+008
20	N	X	X	X	X	X	X	X
20	Y	0.85	6.03E+012	TRUE	153.44	5.469.485	2.07E+008	3.44E+008

Table 4.29: FMS model evaluating the property- E[(M1>0)U(P1s=N and P2s=N and P3s=N)]-New version of the tool without and with the new heuristic algorithm

The results for the models and the properties proposed by the MCC competition are given below.

The following tables show the results for the *FMS model* for all the proposed properties by the MCC.

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.07	2.90E+006	TRUE	0.1	19,588	5.26E+005	1.00E+006
5	Y	0.01	2.90E+006	TRUE	0.02	3,570	9.41E+004	2.16E+005
10	N	0.82	2.50E+009	TRUE	1.1	132,138	4.20E+006	6.67E+006
10	Y	0.08	2.50E+009	TRUE	0.06	9,814	3.10E+005	5.58E+005
15	N	3.45	2.17E+011	TRUE	3.78	414,229	1.51E+007	2.50E+007
15	Y	0.28	2.17E+011	TRUE	0.11	18,433	6.76E+005	1.18E+006
20	N	9.91	6.03E+012	TRUE	10.13	944,965	3.90E+007	5.81E+007
20	Y	0.7	6.03E+012	TRUE	0.22	29,751	1.24E+006	2.02E+006
25	N	23.36	8.54E+013	TRUE	21.61	1,803,184	8.28E+007	1.30E+008
25	Y	1.61	8.54E+013	TRUE	0.37	43,753	2.03E+006	3.46E+006
30	N	49.05	7.74E+014	TRUE	40.36	3,067,639	1.55E+008	2.38E+008
30	Y	2.92	7.74E+014	TRUE	0.6	60,472	3.09E+006	5.11E+006
35	N	86.84	5.09E+015	TRUE	77.75	4,817,109	2.67E+008	3.78E+008
35	Y	5.34	5.09E+015	TRUE	0.86	79,873	4.45E+006	7.07E+006

Table 4.30: FMS model evaluating the non safety property - New version of the tool without and with the new heuristic algorithm

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.08	2.90E+006	FALSE	0.09	10,808	2.93E+005	5.19E+005
5	Y	0.02	2.90E+006	FALSE	0.01	1,468	3.81E+004	1.20E+005
10	N	0.8	2.50E+009	FALSE	0.89	64,367	2.06E+006	3.69E+006
10	Y	0.1	2.50E+009	FALSE	0.04	4,348	1.33E+005	2.90E+005
15	N	3.45	2.17E+011	FALSE	3.56	203,271	7.53E+006	1.33E+007
15	Y	0.3	2.17E+011	FALSE	0.07	9,301	3.27E+005	7.07E+005
20	N	9.91	6.03E+012	FALSE	9.53	437,929	1.83E+007	2.85E+007
20	Y	0.7	6.03E+012	FALSE	0.16	15,798	6.38E+005	1.32E+006
25	N	23.87	8.54E+013	FALSE	21.55	817,226	3.87E+007	6.08E+007
25	Y	1.62	8.54E+013	FALSE	0.25	28,342	1.27E+006	2.58E+006
30	N	49.28	7.74E+014	FALSE	42.22	1,604,184	8.19E+007	1.29E+008
30	Y	2.95	7.74E+014	FALSE	0.41	44,140	2.19E+006	3.80E+006
35	N	87.06	5.09E+015	FALSE	70.91	2,414,959	1.35E+008	2.12E+008
35	Y	5.27	5.09E+015	FALSE	0.55	63,165	3.43E+006	6.27E+006

Table 4.31: FMS model evaluating the deadlock property - New version of the tool without and with the new heuristic algorithm

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.08	2.90E+006	FALSE	0.11	14,629	3.96E+005	7.24E+005
5	Y	0.01	2.90E+006	FALSE	0.01	1,776	4.63E+004	1.33E+005
10	N	0.79	2.50E+009	FALSE	1.11	89,254	2.86E+006	5.11E+006
10	Y	0.08	2.50E+009	FALSE	0.04	4,744	1.46E+005	3.03E+005
15	N	3.55	2.17E+011	FALSE	4.08	245,989	9.17E+006	1.58E+007
15	Y	0.29	2.17E+011	FALSE	0.09	10,525	3.76E+005	7.07E+005
20	N	9.99	6.03E+012	FALSE	11.05	581,844	2.44E+007	3.77E+007
20	Y	0.72	6.03E+012	FALSE	0.21	17,180	6.90E+005	1.33E+006
25	N	24.04	8.54E+013	FALSE	24.5	1,082,733	5.08E+007	8.12E+007
25	Y	1.67	8.54E+013	FALSE	0.35	28,342	1.27E+006	2.58E+006
30	N	50.51	7.74E+014	FALSE	50.12	1,765,746	9.22E+007	1.48E+008
30	Y	3.02	7.74E+014	FALSE	0.5	44,140	2.19E+006	3.83E+006
35	N	87.57	5.09E+015	FALSE	89.81	2,704,339	1.55E+008	2.47E+008
35	Y	5.23	5.09E+015	FALSE	0.83	63,165	3.43E+006	6.27E+006

Table 4.32: FMS model evaluating the reversibility property - New version of the tool without and with the new heuristic algorithm

The results for the *Kanban model* and the properties that were proposed by the MCC competition.

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.01	2.55E+006	TRUE	0.01	1,460	4.08E+004	9.71E+004
5	Y	0.01	2.55E+006	TRUE	0.01	1,268	3.39E+004	8.99E+004
10	N	0.01	1.01E+009	TRUE	0.03	3,705	1.30E+005	2.59E+005
10	Y	0.01	1.01E+009	TRUE	0.03	3,015	9.45E+004	1.85E+005
15	N	0.04	4.70E+010	TRUE	0.04	7,000	2.97E+005	4.59E+005
15	Y	0.02	4.70E+010	TRUE	0.04	5,498	1.95E+005	3.51E+005
20	N	0.09	8.05E+011	TRUE	0.1	11,353	5.67E+005	8.95E+005
20	Y	0.04	8.05E+011	TRUE	0.07	8,738	3.46E+005	6.12E+005
25	N	0.17	7.68E+012	TRUE	0.16	16,740	9.59E+005	1.43E+006
25	Y	0.06	7.68E+012	TRUE	0.08	12,728	5.56E+005	8.94E+005
30	N	0.27	4.99E+013	TRUE	0.23	23,193	1.50E+006	2.15E+006
30	Y	0.09	4.99E+013	TRUE	0.14	17,468	8.31E+005	1.23E+006
35	N	0.43	2.46E+014	TRUE	0.33	30,680	2.21E+006	3.05E+006
35	Y	0.15	2.46E+014	TRUE	0.21	22,967	1.18E+006	1.76E+006

Table 4.33: Kanban model evaluating the non safety property - New version of the tool without and with the new heuristic algorithm

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.01	2.55E+006	FALSE	0.01	471	1.31E+004	7.96E+004
5	Y	0.01	2.55E+006	FALSE	0.01	385	9.92E+003	7.96E+004
10	N	0.01	1.01E+009	FALSE	0.01	1,588	5.50E+004	1.32E+005
10	Y	0.01	1.01E+009	FALSE	0.01	1,086	3.22E+004	9.81E+004
15	N	0.04	4.70E+010	FALSE	0.04	3,264	1.37E+005	2.37E+005
15	Y	0.02	4.70E+010	FALSE	0.02	2,079	6.92E+004	1.43E+005
20	N	0.08	8.05E+011	FALSE	0.09	5,397	2.66E+005	4.59E+005
20	Y	0.04	8.05E+011	FALSE	0.04	3,294	1.22E+005	2.12E+005
25	N	0.17	7.68E+012	FALSE	0.12	6,888	3.83E+005	6.22E+005
25	Y	0.08	7.68E+012	FALSE	0.07	4,249	1.75E+005	3.09E+005
30	N	0.27	4.99E+013	FALSE	0.22	11,887	7.66E+005	1.31E+006
30	Y	0.09	4.99E+013	FALSE	0.07	6,910	3.05E+005	4.77E+005
35	N	0.44	2.46E+014	FALSE	0.35	14,108	9.93E+00	1.35E+006
35	Y	0.17	2.46E+014	FALSE	0.12	8,460	4.01E+005	6.61E+005

Table 4.34: Kanban model evaluating the deadlock property - New version of the tool without and with the new heuristic algorithm

Bound	P-inv	RS Time Gen.	RS Size	Prop.Eval.	Time Eval.	#Nodes	Mem.Use	Mem.All.
5	N	0.01	2.55E+006	TRUE	0.2	18,039	5.11E+005	9.12E+005
5	Y	0.01	2.55E+006	TRUE	0.14	14,030	3.66E+005	7.05E+005
10	N	0.01	1.01E+009	TRUE	2.33	107,863	3.87E+006	6.75E+006
10	Y	0.01	1.01E+009	TRUE	1.22	87,319	2.69E+006	4.45E+006
15	N	0.04	4.70E+010	TRUE	9.14	385,143	1.67E+007	2.65E+007
15	Y	0.03	4.70E+010	TRUE	4.85	231,756	8.04E+006	1.42E+007
20	N	0.07	8.05E+011	TRUE	30.45	807,172	4.03E+007	6.27E+007
20	Y	0.03	8.05E+011	TRUE	12.64	533,923	2.06E+007	3.22E+007
25	N	0.17	7.68E+012	TRUE	70.09	1,611,212	9.31E+007	1.34E+008
25	Y	0.07	7.68E+012	TRUE	26.84	974,320	4.12E+007	5.92E+007
30	X	X	X	X	X	X	X	X
30	Y	0.1	4.99E+013	TRUE	52.59	1,799,068	8.13E+007	1.27E+008
35	N	X	X	X	X	X	X	X
35	Y	0.15	2.46E+014	TRUE	93.92	2,498,388	1.20E+008	1.85E+008

Table 4.35: Kanban model evaluating the reversibility property - New version of the tool without and with the new heuristic algorithm

From the results that we have obtained it is obvious that with the introduction of the new heuristic algorithm and the ordering of the variablesplaces based on the P-invariants we are guaranteed better results in total (time and memory footprint)

Chapter 5

Future work

By exploiting the P-invariants we observed that we can approach a better solution to the MDD ordering, i.e. smaller number of nodes, resulting to smaller memory footprint and better time execution. The next step is to examing the behavior of all different types of Petri Nets systems (*Stochastic Petri Nets*, *Generalized Stochastic Petri Nets*, *Colored Petri Nets*, due to the fact that in this diploma thesis we focused on the study of the *Non Stochastic Petri Nets*. The disadvantage of this particular algorithm is that it can not be performed on Petri Net systems that are not all places covered by the P-invariants. Likewise, our next step is to propose an new heuristic algorithm in order to approach the optimal ordering of the MDDs for Petri Net systems that does not provide us with the information of the P-invariants.

Bibliography

[1] Model checking contest, 2012.

[2] Performance models for discrete events systems with synchronisations : Formalism and analysis techniques.

[3] Sheldon B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, 27(6):509–516, 1978.

[4] Junaid Babar, Marco Beccuti, Susanna Donatelli, and Andrew Miner. Greatspn enhanced with decision diagram data structures. In *Proceedings of the 31st international conference on Applications and Theory of Petri Nets*, PETRI NETS'10, pages 308–317, Berlin, Heidelberg, 2010. Springer-Verlag.

[5] Joao Paulo Barros and Luis Gomes. Actions as activities and activities as petri nets. In *Critical Systems Development with UML: Proceedings of the UML'03 workshop*, pages 129–135, 2003.

[6] M. Ben-Ari. *Principles of concurrent and distributed programming.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[7] Beate Bollig and Ingo Wegener. Improving the variable ordering of obdds is np-cmplete. *IEEE Trans. Comput.*, 45(9):993–1002, sep 1996.

[8] G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. Greatspn 1.7: Graphical editor and analyzer for timed and stochastic petri nets. 24:47–68, 1995.

[9] Gianfranco Ciardo. Data representation and efficient solution: a decision diagram approach. In *Proceedings of the 7th international conference on Formal methods for performance evaluation*, SFM'07, pages 371–394, Berlin, Heidelberg, 2007. Springer-Verlag.

[10] Gianfranco Ciardo, Gerald Lüttgen, and Andy Jinqing Yu. Improving static variable orders via invariants. In *Proceedings of the 28th international conference on Applications and theory of Petri nets and other models of concurrency*, ICATPN'07, pages 83–103, Berlin, Heidelberg, 2007. Springer-Verlag.

[11] Tarik Hadzic, Esben Rune, and Barry O'Sullivan. On Automata Mdds and Bdds in Constraint Satisfaction. 2008.

[12] Alan J. Hu and David L. Dill. Reducing bdd size by exploiting functional dependencies. In *Proceedings of the 30th international Design Automation Conference*, DAC '93, pages 266–271, New York, NY, USA, 1993. ACM.

[13] Intel. Fdiv replacement program: Description of the flaw. Technical report, 2004.

[14] D. Kartson, G. Balbo, S. Donatelli, G. Franceschinis, and Giuseppe Conte. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[15] Vasgen Melikyan. Vlsi verification algorithms. Synopsis University Courseware, 2010.

[16] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[17] Dr. Thomas R. Nicely. Pentium fdiv flaw faq, 1994.

[18] PerformanceEvaluationGroup. *GreatSPN User's Manual 2.0.2.* http://www.di.unito.it/ greatspn/index.html.

[19] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Universität Hamburg, 1962.

[20] Carl Adam Petri. *Communication with automata*. PhD thesis, Universität Hamburg, 1966.

[21] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57, Washington, DC, USA, 1977. IEEE Computer Society.

[22] Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell Systems Technical Journal*, 28:59–98, 1949.

[23] Radu I. Siminiceanu and Gianfranco Ciardo. New metrics for static variable ordering in decision diagrams. In *In TACAS, LNCS 3920:90–104*, pages 328–342. Springer, 2006.

[24] Abhay Vardhan and Mahesh Viswanathan. Learning to verify branching time properties. *Form. Methods Syst. Des.*, 31(1):35–61, aug 2007.