 Electronic Notes in Theoretical Computer Science 98 (2004) 35–55 

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Petri Nets with Non-blocking Arcs are Difficult to Analyze

# Jean-Fran¸cois Raskin [1](#_bookmark8) *,*[2](#_bookmark8) and Laurent Van Begin[3](#_bookmark8) *,*[4](#_bookmark8)

*Universit´e Libre de Bruxelles*

*Blvd du Triomphe, 1050 Bruxelles, Belgium.*

Abstract

In this paper, we study the decidability of five problems on a class of extended Petri nets. The study of this class of extended Petri nets is motivated by the problem of parametric verification of multiple copies of processes that can communicate with a *partially non-blocking rendez-vous*. This kind of communications occurs in abstractions of multi-threaded JAVA programs.

*Keywords:* Monotonic Extensions of Petri Nets, Decidability/ Undecidability.

# Introduction

In parametric verification, we want to verify at once an entire family of sys- tems. For example, some mutual exclusion protocols have been designed to work for any number of (identical) processes. The verification of such pro- tocols for specific number of processes is not satisfactory. We want a proof for any number of those processes. This problem of parametric verification is difficult and has been shown undecidable [[2](#_bookmark11)] in general. To obtain partial automatic methods, several abstractions have been shown useful. The work

1 Email: [jraskin@ulb.ac.be](mailto:jraskin@ulb.ac.be)

2 This author was partially supported by the FRFC grant 2.4530.02.

3 Email: [lvbegin@ulb.ac.be](mailto:lvbegin@ulb.ac.be)

4 This author was supported by a ”First Europe” grant EPH3310300R0012 of the Walloon Region.

1571-0661 © 2004 Elsevier B.V. Open access under [CC BY-NC-ND license.](http://creativecommons.org/licenses/by-nc-nd/3.0/)

doi:10.1016/j.entcs.2003.10.005

in this paper is directly connected to the context of one of these abstractions, the so-called *counting abstraction* [[12](#_bookmark21)].

When considering counting abstractions, (infinite) Petri nets and their ex- tensions are particularly important. In that context, processes of a parametric system are abstracted by tokens, places are used to count the number of pro- cesses in each local state of the parametric system and transitions are used to model the dynamics of the processes. Sistla et al [[12](#_bookmark21)] have shown that Petri nets are well suited to abstract parametric systems where rendez-vous communications are used for synchronizations between processes. When the underlying systems use more “exotic” communication mechanisms, like broad- cast communications for example, the model of Petri nets has to be extended, with transfer arcs for instance, see [[7](#_bookmark17)] for more details.

In this paper, we consider a very simple extension of Petri nets that is able to model parametric systems that uses “partially non-blocking” rendez-vous synchronizations in addition to classical (blocking) rendez-vous synchroniza- tions. Partially non-blocking rendez-vous are asymmetric synchronizations where the sending part is *not blocking* (contrary to the usual case) and the receiving part is *blocking* (as in the usual case). To illustrate the notion of *partially non-blocking rendez-vous*, consider Figure [1](#_bookmark0). This figure represents

*a* ↑ *a* ↓



*l*1

*l*2



*l*3

*l*4

Fig. 1. Example of partially non-blocking rendez-vous.

fragments of two processes. In location *l*1, the first process can emit *a* ↑, the proposition of a rendez-vous on symbol *a* and moves to *l*2 even if the second process is not present to synchronize on *a* by emitting *a* . If the second process can synchronize then it does. On the other hand, the second process has to synchronize with the first process in order to emit *a* ↓ and move from *l*3 to *l*4, it cannot move alone. So the emission *a* ↑ is non-blocking and can occur without a reception part *a* ↓ while the reception is blocking and can only occur in the presence of the emission. This is why we call such rendez-vous “partially non-blocking rendez-vous”. In this paper, we will define a simple extension of the basic Petri nets that is able to model this kind of communi- cations between processes, we call this extension *Petri nets with non-blocking arcs*.

↓

The study of this simple extension of Petri nets is motivated by previous works by the authors on extensions of Petri nets for modeling communications in multi-threaded programs [[7](#_bookmark17)]. Multi-threaded JAVA programs use instruc- tions like notify and notifyAll for synchronizations. The instruction notify can be modeled by an partially non-blocking rendez-vous and the instruction

notifyAll can be modeled by a broadcast communication. While transfer nets, that are able to model broadcast communications, have been studied from a theoretical point of view [[9](#_bookmark19),[5](#_bookmark15)], this is not the case for extensions of Petri nets modeling partially non-blocking rendez-vous. We study here the decidability of five important problems in the context of Petri nets extended with non-blocking arcs. While those five problems are decidable for Petri nets, we show here that only two of them remain decidable in the extended model. The rest of the paper is organized as follows. In a second section, we recall some basic notions and notations. In a third section, we introduce Petri nets extended with non-blocking arcs and the five problems that we study. In a fourth section, we show that two of the problems remain decidable on the extended model. In a fifth section, we establish the undecidability of the three

other problems.

Finally, a last section draws some conclusions.

# Preliminaries

Multi-sets.

A multi-set *B* constructed from a set *S* of *n* elements is a function *B* : *S* → N that assigns to each element *s* of *S* the number *B*(*s*) of occurrences of *s* in the multi-set. To denote a multi-set *S* over *S* = {*s*1*,... , sn*}, we write

{(*si, B*(*si*))|*B*(*si*) *>* 0}. For example, let *S* be {*s*1*, s*2*, s*3} and let *B* be such that *B*(*s*1)= 3, *B*(*s*2)= 0, *B*(*s*3)= 1, then *B* is denoted by (*s*2*,* 3)*,* (*s*3*,* 1) . Equivalently, *B* can be represented as a *n*-dimensional vector, denoted vec(*B*),

{ }

and defined as follows:

 *B*(*s*1) 

vec(*B*)=  *B*(*s*2) 

 *...* 

 *B*(*sn*) 

Well quasi orderings, well structured transition systems.

A *well quasi ordering* “ on the elements of a set *S* is a reflexive and transitive relation such that for any infinite sequence *s*1*s*2 *...* where *si* ∈ *S* (*i* ≥ 1) there is *i < j* such that *si* “ *sj*. In the following we note *si* ≺ *sj* if *si* “ *sj* but *sj* “ *si*. As an example, it is well know that the quasi order “ on elements in N*k* defined as *m* “ *m*' if *mi m*' for any 1 *i k* is a well quasi ordering. In the rest of this paper, we consider this quasi order on vectors of naturals.

*i*

≤ ≤ ≤

/

A *transition system* is a tuple ⟨*L,* →⟩ where *L* is a set of states and →⊆ *L* × *L*. ⟨*l*1*, l*2⟩ ∈→ is noted *l*1 → *l*2. A transition system ⟨*L,* →⟩ is *monotonic* according to the well quasi ordering “ on the elements of *L* if for all *l*1*, l*2 in

*L* with *l*1 “ *l*2, if *l*1 → *l*' then there exists *l*' *l*' with *l*2 → *l*' . A transition

1

2

1

2

system ⟨*L,* →⟩ is *strictly monotonic* according to the well quasi ordering “

on the elements of *L* if it is monotonic and for all *l*1*, l*2 in *L* with *l*1 ≺ *l*2, if

*l*1 → *l*' then there exists *l*' > *l*1 with *l*2 → *l*' . Systems that are monotonic for

1

2

'

2

a well-quasy order “ are called *well structured transition systems* in [[11](#_bookmark22)]. For

those systems, several general decidability results are known [[1](#_bookmark12),[11](#_bookmark22)]. We will use those results in section 4 to derive the decidability of two problems on our extended model of Petri nets.

A *two-counter machine C*, 2CM for short, is a tuple ⟨*c*1*, c*2*, L,* Instr⟩ where:

* *c*1, *c*2 are two counters taking their values in N;
* *L* = {*l*1*, l*2*,... , lu*} is a finite non-empty set of *u* locations;
* Instr is a function that labels each location *l L* with an instruction that has one of the three following forms:

∈

· *l* : *cj* := *cj* + 1; goto *l* ;, where *j* ∈ {1*,* 2} and *l*

'

'

increment, and we define TypeInst(*l*)= incj;

'

∈ *L*, this is called an

'

· *l* : *cj* := *cj* − 1; goto *l* ;, where *j* ∈ {1*,* 2} and *l*

decrement, and we define TypeInst(*l*)= decj;

∈ *L*, this is called a

' '' ' ''

· *l* : if *cj* = 0 then goto *l* else goto *l* ;, where *j* ∈ {1*,* 2} and *l ,l*

this is called a zero-test, and we define TypeInst(*l*)= zerotestj.

∈ *L*,

Those instructions have their usual obvious semantics, in particular, decre- ment can only be done if the value of the counter is strictly greater than zero.

A *configuration* of a 2CM ⟨*c*1*, c*2*, L,* Instr⟩ is a tuple ⟨*loc, v*1*, v*2⟩ where *loc* ∈ *L* is the value of the program counter and *v*1, respectively *v*2, is a natural number that gives the valuation of the counter *c*1, respectively *c*2. A *computation γ* of a 2CM ⟨*c*1*, c*2*, L,* Instr⟩ is either a finite sequence of config- urations ⟨*loc*1*, v*1*, v*2⟩*,* ⟨*loc*2*, v*1*, v*2⟩*,... ,* ⟨*locr, v*1*, v*2⟩, or an infinite sequence of

1

1

2

2

*r*

*r*

configurations ⟨*loc*1*, v*1*, v*2⟩*,* ⟨*loc*2*, v*1*, v*2⟩ *,... ,* ⟨*locr, v*1*, v*2⟩*,...* such that : (*i*)

1

1

2

2

*r*

*r*

“Initialization”: *loc*1 = *l*1, *v*1 = 0, and *v*2 = 0, i.e. a computation starts in *l*1

1 1

and the two counters have the value zero; (*ii*) “Consecution”: for each *i* ∈ N

such that 1 ≤ *i* ≤ |*γ*| we have that ⟨*loci*+1*, v*1

*i*+1

*, v*

2

*i*+1

⟩ is the configuration ob-

tained from ⟨*loci, v*1*, v*2⟩ by applying the instruction Instr(*loci*). In the finite

*i*

*i*

case, *r* is the *length* of the computation *γ* and we define final(*γ*)= ⟨*locr, v*1*, v*2⟩.

*r*

*r*

If *γ* is a computation, *γi* denotes the *ith* configuration of *γ*. A configuration

⟨*loc, v*1*, v*2⟩ is *reachable* in the 2CM ⟨*c*1*, c*2*, L,* Instr⟩, if there exists a finite com- putation *γ* such that final(*γ*)= ⟨*loc, v*1*, v*2⟩.

The *reachability problem for* 2CM is defined as follows: “Given a 2CM *C* =

⟨*c*1*, c*2*, L,* Instr⟩ and a configuration ⟨*loc, v*1*, v*2⟩ of *C*, is ⟨*loc, v*1*, v*2⟩ reachable from ⟨*l*1*,* 0*,* 0⟩ ?”. The *boundedness problem for* 2CM is defined as follows: “Given a 2CM *C* = ⟨*c*1*, c*2*, L,* Instr⟩, is there *c* ∈ N such that for all reachable configuration ⟨*loc, v*1*, v*2⟩ in *C* we have *v*1 + *v*2 ≤ *c* ?”

It is well-known that those two problems cannot be answered completely with an algorithm.

Theorem 2.1 (From [[16](#_bookmark26)]) *The reachability and boundedness problems are undecidable for* 2CM*.*

# Petri nets extended with non-blocking arcs

In this section, we introduce formally the class of extended Petri nets that we call *Petri nets with non-blocking arcs*.

Definition 3.1 A *Petri Net with non-blocking arcs* N , PN+NBA for short, is defined by a pair N = ⟨P*,* T⟩ where P = {*p*1*, p*2*,... , pn*} is a finite set of *n* places and T = {*tr*1*, tr*2*,... , trm*} is a finite set of *m* transitions where each *tri* ∈ T is a tuple ⟨*I, O, A*⟩, where *I* is a multi-set of input places in P, *O* is a multi-set of output places in P, and *A* the non-blocking part of the transition is either the empty set or a singleton {⟨*p, q*⟩} with *p, q* ∈ P \ {*r* | (*r, i*) ∈ or ( ) and 1 , and are called respectively the source and the

∈ ≥ }

*I r, i O, i p q*

target of the non-blocking part.

A marking of a PN+NBA N = ⟨P*,* T⟩ is a function *m* : P → N that assigns to each place a natural number ( ). Equivalently, a marking *m* can be seen as a *n*-dimensional vector of natural numbers. In the following, for a marking *m* and a set of places *S*, we will write *m*(*S*) for *p*∈*S m*(*p*).

Σ

∈ P

*p m p*

Figure [2](#_bookmark1) shows an example of PN+NBA. Circles represent places and filled

rectangles represent transitions. Plain edges from places *p* to transitions *tr* are labeled by the number of occurrences of *p* in the input multi-set of *tr* and plain edges from transitions *tr* to places *p* are labeled by the number of occurrences of *p* in the output multi-set of *tr*. Absence of edge from (to) a place *p* to (from) a transition *tr* means that there is no occurrence of *p* in the input (output) multi-set of *tr*. In the following, when there is only one occurrence of a place into a given multi-set of a transition we will only use edges without labels. Pairs of dashed edges from a place to a transition and from this transition to a place represent the non-blocking part of the transition. Tokens in the places define markings in the usual way.

A transition *tr* = *I, O, A* is *firable* in a marking *m* iff *m* vec(*I*). Note that the non-blocking part is not taken into account to decide if a transition *tr* is firable in a marking *m* or not. Given a marking *m* and a transition

⟨ ⟩

*tr* = ⟨*I, O, A*⟩ that is firable in *m*, we say that *m* leads to *m*' by firing *tr*, noted *m* →*tr m*' where *m*' is defined as:

* if *A* = {⟨*p, q*⟩} and *m*(*p*) ≥ 1: *m*' = *m* − vec(*I*)+ vec(*O*) − vec({(*p,* 1)})+ vec({(*q,* 1)}), that is the input places are decremented by their number of occurrences in , the output places are incremented by their number of occurrences in *O* and one token moves from the source place to the target place of the non-blocking part.

*I*

* otherwise: *m*' = *m* vec(*I*)+ vec(*O*). In that case, either there is no non- blocking part to the transition and the effect of the transition is as in the usual Petri net case or the source of the non-blocking part *p* is not marked and the non-blocking part has no effect.

—

A computation *η* of a PN+NBA N = ⟨P*,* T⟩ is a sequence of markings al- ternating with transitions *η* = *m*1*tr*1*m*2*tr*2 *... trr*−1*mr* where *mi* is a marking for any *i* ∈ {1*,* 2*,..., r*}, *trj* ∈ T for any *j* ∈ {1*,* 2*,...,r* − 1} and we have that *m*1 →*tr*1 *m*2 →*tr*2 *...* →*trr*−1 *mr*. This notion of computation is extended to the infinite case as usual. A sequence of transitions *σ* = *tr*1*tr*2 *... trr* is firable in a marking *m*1 if there exists a sequence of markings *m*1*m*2 *... mr*+1 such that *m*1*tr*1*m*2*tr*2 *... trrmr*+1 is a computation of . We note *m σ m*' the fact that firing *σ* from *m* leads to *m*'. A marking *m*' is *reachable* from a marking *m* in N iff there exists a sequence of transitions *σ* of N such that *m* →*σ m*'. We note Reach(N *, m*) the set of markings that are reachable from *m* in N , i.e. Reach(N *, m*)= {*m*'|∃*σ* ∈ T ∗ : *m* →*σ m*'}.

N →

A labeled PN+NBA is a tuple *, ,* where and are a set of places and a set of transitions as before and : Σ is a labeling function that labels each transition *tr* with the label (*tr*) from a finite set of labels Σ. The notion of computation is as before. To each of those com- putations *η* = *m*1*tr*1*m*2*tr*2 *... mrtrr ...* we associate the sequence of labels L(*η*) = L(*tr*1)L(*tr*2) *...* L(*trn*) *...* For a PN+NBA N and a marking *m*, we define L(N *, m*)= {L(*η*)|*η* is an infinite computation of N with initial mark- ing . The formula of the logic are evaluated over those sequences of labels. Given a set of labels Σ, the formulas of the logic LTL are defined by the following rule:

}

∈ T L

L T →

*m* LTL

⟨P T L⟩ P T

*φ* := *λ*|¬*φ*|*φ*1 ∨ *φ*2|◯ *φ*| *φ*| *φ*|*φ*1U *φ*2

where *λ* Σ. We only give the semantics for the and operators because they are the only ones that we need in this paper. For Λ Σ*ω* such that Λ = *λ*1 *... λiλi*+1 *.. .*, we note Λ*i* for the suffix *λiλi*+1 *...* of Λ starting at this index *i* and Λ(*i*) for for the *ith* element in Λ. Given Λ ∈ Σ*ω* and a formula *φ*, we define the satisfaction relation, noted |=, as follows :

∈

∈

* if *φ* = *λ*, then Λ |= *φ* iff Λ(1) = *λ*;
* if *φ* = *ϕ*, then Λ |= *φ* iff ∃*i* ≥ 1: Λ*i* |= *ϕ*;
* if *φ* = *ϕ*, then Λ |= *φ* iff ∀*i* ≥ 1: Λ*i* |= *ϕ*.

For a set of infinite sequence of labels *M* and a formula *φ*, we have *M* |= *φ* if for all Λ ∈ *M* we have Λ |= *φ*.

*p*4



1

*p*2

1 1

*t*2

*t*1

1

1 1

*p*1 *p*3

Fig. 2. a Petri net with non-blocking arcs.

The PN+NBA of Figure [2](#_bookmark1) has two transitions. The transitions *t*1 is a classical Petri net transition while *t*2 has an non-blocking part. Let us make the hypothesis that the tokens represent processes and places represent local states of processes. In that context, transition *t*1 models an usual rendez-vous

: “If one process is in its local state *p*1 and another is its local state *p*3, then the two processes can synchronize and move synchronously to their local states *p*2 and *p*4, respectively”. In the same context the transition *t*2 models a “partially non-blocking” rendez-vous : “If there is one process in *p*4 and one in *p*2, then the two processes can synchronize and move to *p*1 and *p*3 respectively. If no process are present in *p*4, a process in *p*2 does not have to wait and can move to its local state *p*1”. In that context, the process in *p*2 proposes a rendez-vous to processes in *p*4. If at least one process is present in *p*4 the rendez-vous takes place, otherwise the process in *p*2 does not have to wait and can proceed.

Problems

The *marking reachability problem* for a PN+NBA , is the problem defined as follows: “Given a PN+NBA with an initial marking *m* and a marking *m*', does *m*' belong to Reach( *, m*) ?”. The *marking coverability problem* for a PN+NBA is the problem defined as follows: “Given a PN+NBA with an initial marking *m* and a marking *m*', does there exist a marking *m*'' that belongs to Reach(N *, m*) and such that *m*' “ *m*'' ?”. The *boundedness problem* for a PN+NBA N is the problem defined as follows : “Given a PN+NBA N and an initial marking *m*, is Reach(N *, m*) finite ?”. The *place boundedness problem* for a PN+NBA N is the problem defined as follows : “Given a PN+NBA N , an

N N

N

N

N

initial marking *m* and a place *p*, is there *c* ∈ N such that ∀*m*' ∈ Reach(N *, m*) we have *m*'(*p*) ≤ *c* ?” The *action-based* LTL *model checking problem* for a labeled PN+NBA N is the problem defined as follows : “Given a labeled PN+NBA N , an initial marking *m* and an action-based LTL formula *φ*, does L(N *, m*) |= *φ* hold ?”

It is well-known that those five problems are decidable on Petri nets [[13](#_bookmark23),[14](#_bookmark24),[10](#_bookmark20)].

Theorem 3.2 *The marking reachability, marking coverability, boundedness, place boundedness and action-based* LTL *model checking problems are decidable on Petri nets.*

In the next sections we will investigate the decidability of those problems for PN+NBA.

# Decidability results

We give here two positive algorithmic results for the analysis of PN+NBA. They are a direct consequence of the strict monotonicity property of that class of extended Petri nets.

Proposition 4.1 *The class* PN+NBA *is strictly monotonic.*

From Proposition [4.1](#_bookmark2) and [[1](#_bookmark12),[11](#_bookmark22)], we deduce the decidability of the cover- ability problem and the boundedness problem for PN+NBA.

Corollary 4.2 *The coverability problem and the boundedness problem are de- cidable for the class* PN+NBA*.*

# Undecidability results

In the previous section, we have seen that the coverability problem and the boundedness problem are decidable for PN+NBA. In this section we show that all the other problems that are decidable for Petri nets become undecidable for PN+NBA.

To establish those undecidability results, we will show that PN+NBA are able to partially simulate the computations of a 2CM. This partial simulation result will allow us to reduce in an uniform way undecidable problems for 2CM to problems for PN+NBA.

*li* : *cj* := *cj* + 1; goto *l*' *li* : *cj* := *cj* − 1; goto *l*'

*t*

*li* : if *ci* =0 then goto *l*' else goto *l*''

*ti ti*

*li*

*l*'

=0 /=0

*i* *i*

*li*

*l*''

*li*

*l*'

*li*

*l*'

*t*

*cj K*

*cj K*

*cj T cj*

(*a*) (*b*) (*c*)

Fig. 3. Simulation of the operations of a 2CM by PN+NBA transitions.

* 1. *Partial simulations of a* 2CM *by a* PN+NBA

Widget.

For any 2CM *C* = ⟨*c*1*, c*2*,L* = {*l*1*, l*2*,... , lu*}*,* Instr⟩, we construct a Petri net with non-blocking arcs N*C* = ⟨P*,* T ⟩, called the *simulation widget*, defined as follows. The set of places P is equal to {*c*1*, c*2*, l*1*, l*2*,... , lu, K,T* }. The places *c*1 and *c*2 will be used to keep track of the values of the two counters of *C*, *l*1*, l*2*, , lu* called the *control places* will be used to keep track of the program counter of *C*, *K* is called the *capacity place*, *T* is called the *trash*. The use of *K* and *T* will be described below. The set of transitions T is the smallest set of transitions such that for each *li* ∈ *L*:

···

* + - if Instr(*li*) is of the form *cj* := *cj* + 1; goto *l*', then T contains the transition

*tri* = ⟨*I, O, A*⟩ with *I* = {(*li,* 1)*,* (*K,* 1)}, *O* = {(*cj,* 1)*,* (*l ,* 1)}, and *A* = ∅.

'

* + - if Instr(*li*) is of the form *cj* := *cj* − 1; goto *l* , then T contains the transition

'

*tri* = ⟨*I, O, A*⟩ with *I* = {(*li,* 1)*,* (*cj,* 1)}, *O* = {*l , K*}, and *A* = ∅;

'

* + - if Instr(*li*) is of the form if *cj* = 0 then goto *l*' else goto *l*'' then con- tains two transitions *tr*=0 and *tr*/=0 defined as:

T

*i* *i*

· *tr*=0 = ⟨*I, O, A*⟩ with *I* = {(*li,* 1)}, *O* = {(*l*'*,* 1)}, and *A* = {⟨*cj,T* ⟩}.

*i*

/=0 ''

· *tri* = ⟨*I, O, A*⟩ with *I* = {(*li,* 1)*,* (*cj,* 1)}, *O* = {(*cj,* 1)*,* (*l ,* 1)}, and *A* = ∅.

Figure [3](#_bookmark3)(a) shows the transition that simulates an increment of *cj* by mov- ing one token from the capacity place to *cj*. Figure [3](#_bookmark3)(b) shows the transition that simulates a decrement of *cj* by moving one token from *cj* to the capacity place. Figure [3](#_bookmark3)(c) shows the transitions that simulates a zero-test on *cj* when *cj* is equal to zero (transition *t*=0) and when *cj* is greater than zero.

*i*

We note *mk* the marking of the places in P = {*c*1*, c*2*, l*1*, l*2*,... , lu, K,T* } defined as follows: *mk*(*l*1)= 1, for any *l* ∈ {*l*2*, l*3*,... , lu*}, *mk*(*l*)= 0, *mk*(*c*1)= 0, *mk*(*c*2)= 0, *mk*(*K*)= *k*, and *mk*(*T* )= 0.

Properties of the widget.

Let *C* = ⟨*c*1*, c*2*, L,* Instr⟩ be a 2CM and N*C* = ⟨P*,* T⟩ be the simulation wid- get associated to *C* as defined above. Let *γ* = ⟨*loc*1*, v*1*, v*2⟩⟨*loc*2*, v*2*, v*2⟩ *...* be

1

1

2

2

the computation of *C*. We associate to *γ* a sequence of transitions *tr*1*tr*2 *...* of



*K*

*T*

*l*1

*c*1

*lu*

*c*2

(a)

*β*2



*p*1

*v*1 + *v*2

*β*

*p*3

*β*4

5

*β*1

*K*

*T*

*l*1

*p*2

*v*

1

*c*1

*v*2

*β*3

*c*2

*loc*

*p*2 *β*3

(b)



*p*1

*β*2

*β*1

*K*

*T*

*l*1

*c*1

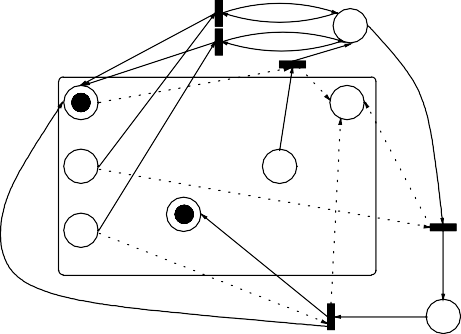
*β*4

*loc*

*c*2

*β*1

*β*3



*β*2

*βli*

*p*1

*K*

*T*

*c*1

*li*

*l*1

*c*2

(c)

*β*4 *p*2

(d)

Fig. 4. Construction using the widget.

N*C*, such that for all *i* ∈ N such that 1 ≤ *i* ≤ |*γ*|, we have *tri* = *α*(⟨*loci, v*1*, v*2⟩)

where *α* is defined as:



*i i*

*α*(⟨*loc, v*1*, v*2⟩)=

*trk* if *loc* = *lk* and TypeInst(*loc*) /= zerotestj



*tr*=0 if *loc* = *lk* and TypeInst(*loc*)= zerotestj and *vj* = 0.

*k*

 *tr*/=0 if *loc* = *lk* and TypeInst(*loc*)= zerotestj and *vj >* 0.

*k*

The sequence of transitions corresponding to *γ* is denoted by *α*(*γ*). The func- tion *α*−1 on the transitions of the simulation widget is defined as:

*α*−1(*tri*)= *li* if *α*(⟨*li, v*1*, v*2⟩)= *tri* for some *v*1*, v*2 ∈ N.

*α*−1 applied on a sequence of transitions *σ* = *tr*1 *... trn* of the widget that is firable from *mk* (*k* ≥ 1), returns a sequence of configurations of *C γ* =

1 2 1 2 1 2 1 2

⟨*loc*0*, v*0 *, v*0 ⟩ ⟨*loc*1*, v*1 *, v*1 ⟩ *...* ⟨*locn, vn, vn*⟩ such that (*i*) *loc*0 = *l*1, *v*0 = 0, *v*0 =0

and (*ii*) for all 1 ≤ *i* ≤ *n*, either TypeInstr(*li*−1) /= *zerotestj* and ⟨*loci, v*1*, v*2⟩

*i*

*i*

is constructed from ⟨*loci*−1*, v*1

*i*−1

*, v*

2

*i*−1

⟩ applying Instr(*li*−1). Or Inst(*li*−1) is of

the form if *cj* =0 then goto *l*' else goto *l*'' and the following cases holds.

* + - *tri* = *tr*=0, then *loci* = *l*', *v*1 = *v*1 and *v*2 = *v*2

, or

*i i*−1 *i i*−1

* + - *tri* = *tr*/=0 and *loci* = *l*'', *v*1 = *v*1 , *v*2 = *v*2 .

*i i*−1 *i i*−1

We now formalize important properties of the widget by the following lemmas. The proofs of those lemmas are easy but tedious and so given in appendix.

Lemma 5.1 *Let γ* = ⟨*loc*1*, v*1*, v*2⟩⟨*loc*2*, v*1*, v*2⟩ *...* ⟨*locr, v*1*, v*2⟩ *be a computa-*

1

1

2

2

*r*

*r*

*tion of the* 2CM *C* = ⟨*c*1*, c*2*, L,* Instr⟩ *such that for any i* ∈ {1*,* 2*,..., r*}*, v*1 + *v*2 ≤ *k. Let* N*C be the simulation widget associated to C. The sequence*

*i*

*i*

*of transitions α*(*γ*) *is firable from the marking mk and firing this sequence of*

*transitions leads to a marking m*' *defined as follows: m*'(*l*) = 1*, for l* = *locr, m*'(*l*') = 0 *for any l*' /= *locr, m*'(*c*1) = *v*1*, m*'(*c*2) = *v*2*, m*'(*K*) = *k* − *v*1 − *v*2*,*

*and m*'(*T* )= 0*.*

*r r r r*

Proof. Given in appendix.

This lemma formalizes the fact that any computation of a 2CM on which the sum of counters does not exceed *k* can be faithfully simulated by its associated widget from marking *mk* with a computation that does not put tokens in *T* .

Lemma 5.2 *Let σ* = *tr*1*tr*2 *... trn be a sequence of transitions of the simula- tion widget C associated to the* 2CM *C* = *c*1*, c*2*, L,* Instr *. If mk σ m*' *and m*'(*T* )= 0*, then α*−1(*σ*) *is a computation of C with* final(*α*−1(*σ*)) = *loc, v*1*, v*2 *such that m*'(*loc*)= 1*, v*1 = *m*'(*c*1) *and v*2 = *m*'(*c*2)*.*

⟨ ⟩

N ⟨ ⟩ →

Proof. Given in appendix.

This second lemma says that any computation of the widget from its initial marking that does not put tokens in *T* is a simulation of a computation of its associated 2CM.

Lemma 5.3 *Let* N*C be the simulation widget associated to the* 2CM *C* =

⟨*c*1*, c*2*, L,* Instr⟩*. For any marking m such that m* ∈ Reach(N*C, mk*)*, we have that m*({*c*1*, c*2*, K,T* })= *k.*

Proof. Given in appendix.

This last lemma says that in any reachable marking of the widget, the sum of the tokens in the set of places {*c*1*, c*2*, K,T* } stays constant.

* 1. *Undecidability proofs*

We are now equipped to establish the undecidability of the marking reach- ability, action-based LTL model checking and place boundedness problems.

Theorem 5.4 *The marking reachability problem is undecidable for* PN+NBA*.*

Proof. Let *C* = ⟨*c*1*, c*2*, L,* Instr⟩ be a 2CM and let *s* = ⟨*loc, v*1*, v*2⟩ be a con- figuration of . Let us show that we can reduce the reachability problem of *s* in *C* to the reachability problem between two markings in a PN+NBA.

*C* [5](#_bookmark9)

We construct the PN+NBA N ' = ⟨P'*,* T '⟩ starting from the simulation widget N*C* = ⟨P*,* T⟩ associated to *C*. To the simulation widget, we add the places and transitions as indicated in figure [4](#_bookmark4)(b). That is, P = P ∪

'

{*p*1*, p*2}, T = T ∪{*β*1*, β*2*, β*3*, β*4} and the new transitions are defined as follows:

'

*β*1 = ⟨*I, O, A*⟩ such that *I* = {(*p*1*,* 1)}, *O* = {(*K,* 1)*,* (*p*1*,* 1)}, and *A* = ∅;

*β*2 = ⟨*I, O, A*⟩ such that *I* = {(*p*1*,* 1)}, *O* = {(*l*1*,* 1)}, and *A* = ∅; *β*3 =

⟨*I, O, A*⟩ such that *I* = {(*loc,* 1)}, *O* = {(*p*2*,* 1)} and *A* = ∅; *β*4 = ⟨*I, O, A*⟩ such that *I* = {(*K,* 1)*,* (*p*2*,* 1)}, *O* = {(*p*2*,* 1)}, and *A* = ∅. We consider the initial marking *m* such that *m*(*p*1) = 1 and for all *p* ∈ P' \ {*p*1}, *m*(*p*) = 0. Furthermore, we consider the marking *ms* defined from the configuration *s* as follows: *ms*(*p*1) = 0, *ms*(*p*2) = 1, *ms*(*l*) = 0 for any *l L*, *ms*(*c*1) = *v*1, *ms*(*c*2) = *v*2, *ms*(*K*) = 0, and *ms*(*T* ) = 0. Let us now show that (*i*) *s* is reachable in *C* iff (*ii*) *ms* is reachable from *m* in N '.

∈

(*i*) → (*ii*). If *s* is reachable in *C* then there exists a computation *γ* =

⟨*loc*1*, v*1*, v*2⟩*,* ⟨*loc*2*, v*1*, v*2⟩*, ... ,* ⟨*locr, v*1*, v*2⟩ with *s* = ⟨*locr, v*1*, v*2⟩. Let us note

1

1

2

2

*r*

*r*

*r*

*r*

*k* the maximum of *c*1 + *c*2 along *γ*. Let us show that we can fire the sequence of transitions *σ* = *βkβ α*(*γ*)*β βk*−*v*1−*v*2 and that *m* →*σ m* . By firing *βkβ* , we

1

2

3

4

*r*

*r*

*s*

1

2

put *k* tokens in the capacity place *K* and one token in control place *l*1. The widget, following Lemma [5.1](#_bookmark5), is now ready to simulate faithfully *γ* by firing the

sequence of transitions *α*(*γ*) as *K* contains enough tokens. As the simulation was faithful, the place *c*1 contains *v*1 tokens and the place *c*2 contains *v*2 tokens.

*r r*

We also know that the place *T* contains no tokens, and so by Lemma [5.3](#_bookmark7) the

place *K* contains *k* − *v*1 − *v*2 tokens. After we can fire *β*3, the control token is

*r*

*r*

moved from the control location *locr* of the widget to the place *p*2. So firing

*k*−*v*1−*v*2

*β*4 *r r* leads to the marking *ms*.

(*ii*) (*i*). Let us make the hypothesis that *ms* is reachable in ' with

→ N

a sequence of transitions *σ* from *m*. Let us show that *σ* must be of the form

5 In the case of reachability, we may simplify a little bit the construction of the widget by suppressing the capacity place *K*. However, to keep the proofs uniform and in particular to be able to use lemmas [5.1](#_bookmark5), [5.2](#_bookmark6) and [5.3](#_bookmark7) in all our proofs, we have decided to keep the widget in its full version for this proof.

*β*∗*β*2*σ*0*β*3*β*∗, where *σ*0 are transitions of the widget. In *m*, *β*1 and *β*2 are the

1 4

only firable transitions. Once *β*2 is fired, place *l*1 is marked and the transitions

*σ*0 of the widget has to be fired. To put one token in *p*2, transition *β*3 has to be fired. After firing *β*3, *β*4 is the only firable transition. It remains us to prove that *α*−1(*σ*0) is a computation of the 2CM *C* that reaches *s*. As *ms*(*T* ) contains no token, by Lemma [5.2](#_bookmark6), we know that the simulation was faithful and so *α*−1(*σ*) leads to *s* in *C*.

Theorem 5.5 *The action-based* LTL *model checking problem is undecidable for labeled* PN+NBA*.*

Proof. Let *C* = *c*1*, c*2*, L,* Instr be a 2CM and let *s* = *loc, v*1*, v*2 be a con- figuration of *C*. Let us show that we can reduce the reachability problem of *s* in *C* to the action-based LTL model checking problem for a PN+NBA.

⟨ ⟩ ⟨ ⟩

We construct the PN+NBA N ' = ⟨P'*,* T '⟩ starting from the simulation widget N*C* = ⟨P*,* T⟩ associated to *C*. To the simulation widget, we add the places and transitions as indicated in figure [4](#_bookmark4)(c). That is, P' = P∪{*p*1*, p*2*, p*3},

T = T ∪ {*β*1*, β*2*, β*3*, β*4*, β*5} and the new transitions are defined as follows:

'

*β*1 = ⟨*I, O, A*⟩ such that *I* = {(*p*1*,* 1)}, *O* = {(*K,* 1)*,* (*p*1*,* 1)}, and *A* = ∅; *β*2 =

⟨*I, O, A*⟩ such that *I* = {(*p*1*,* 1)}, *O* = {(*l*1*,* 1)}, and *A* = ∅; *β*3 = ⟨*I, O, A*⟩ such that *I* = {(*c*1*, v*1)*,* (*c*2*, v*2)*,* (*loc,* 1)}, *O* = {(*p*2*,* 1)}, and *A* = ∅; *β*4 = ⟨*I, O, A*⟩

such that *I* = {(*p*2*,* 1)}, *O* = {(*p*3*,* 1)}, and *A* = {⟨*c*2*,T* ⟩}; *β*5 = ⟨*I, O, A*⟩

such that *I* = {(*p*3*,* 1)}, *O* = {(*l*1*,* 1)*,* (*K, v*1 + *v*2)}, and *A* = {⟨*c*1*,T* ⟩}. The labeling function L is the identity function, that is for any *tr* ∈ T ' we have L(*tr*) = *tr*. We consider the initial marking *m* such that *m*(*p*1) = 1 and for all *p* ∈ P' \{*p*1}, *m*(*p*) = 0. Furthermore, we consider the marking *ms* defined from the configuration *s* as follows: *ms*(*p*1) = 0, *ms*(*p*2) = 0, *ms*(*p*3) = 0, *ms*(*loc*) = 1, *ms*(*l*) = 0 for any *l* /= *loc* ∈ *L*, *ms*(*c*1) = *v*1, *ms*(*c*2) = *v*2, *ms*(*K*) = 0, and *ms*(*T* ) = 0. Let us now show that (*i*) *s* is reachable in *C* iff (*ii*) L(N '*, m*) |= ¬ *β*3. (*i*) → (*ii*). If *s* is reachable in *C* then there exists a computation *γ* = ⟨*loc*1*, v*1*, v*2⟩*,* ⟨*loc*2*, v*1*, v*2⟩*, ... ,* ⟨*locr, v*1*, v*2⟩ with *s* =

1

1

2

2

*r*

*r*

⟨*locr, v*1*, v*2⟩. Let us note *k* the maximum of *c*1 + *c*2 along *γ*. We now construct

*r*

*r*

from *γ* a computation *σ* of N ' such that *σ* |= *β*3. We extend the markings

*mk* (*k* ≥ 1) to P' such that *mk*({*p*1*, p*2*, p*3}) = 0. The sequence of transitions

*α*(*γ*) is such that *m* →*βkβ*2 *m*

→ *m*

1

*k*

*s*

*k*

1

2

*α*(*γ*)

→*β*3*β*4*β*5 *m* . By firing *βkβ* , we

put *k* tokens in the capacity place *K* and one token in the control place *l*1 to reach the marking *mk*. The widget, following Lemma [5.1](#_bookmark5), is now ready to simulate faithfully *γ* leading to *ms* by firing the sequence of transitions *α*(*γ*) as *K* contains enough tokens. After firing *β*3, the control token is moved from the control location *loc* of the widget to the place *p*2, *v*1 tokens are removed from *c*1 and *v*2 tokens are removed from *c*2. Firing *β*4*β*5 moves the control token from *p*2 to *l*1 passing through *p*3 and puts *v*1 + *v*2 into *K* leading to *mk*.

We conclude that the infinite sequence of transitions *σ* = *βkβ*2(*α*(*γ*)*β*3*β*4*β*5)*ω*

1

is firable from *m* and satisfies the formula *β*3 and so L(N *, m*) |= ¬ *β*3.

(*ii*) (*i*). Let us make the hypothesis that there is a sequence of labels as- sociated to a computation of ' from the marking *m* and satisfying the formula

N

→

*β*3. Let us show that the infinite sequence of transitions *σ* corresponding to such a computation must be of the form *β*∗*β*2*σ*0*β*3*β*4*β*5 *... σnβ*3*β*4*β*5 *.. .*, where each *σi*(*i* ≥ 0) is a sequence of transitions of the widget. In fact, *β*1 and *β*2 are the only firable transitions from *m*. Once *β*2 is fired, place *l*1 is marked and a sequence of transitions of the widget *σ*0 must be fired. After firing *β*3, *β*4 followed by *β*5 are the only firable transitions, then a sequence of transitions of the widget *σ*1 must be fired, etc.

1

Suppose that *s* is not reachable and let us derive a contradiction. Assume that we have *m*1 →*σ*1 *...* →*β*3*β*4 *β*5 *m*2*i*−1 →*σi m*2*i* →*β*3*β*4*β*5 *m*2*i*+1 →*σi*+1 *...* For each *i* ≥ 1, two cases are possible:

1. *m*2*i*−1(*c*1)= *m*2*i*−1(*c*2) = 0. We consider here two subcases.
   * (1*a*) *m*2*i*(*c*1) = *v*1 and *m*2*i*(*c*2) = *v*2. As we suppose that *s* is not reachable, we have that *α*−1(*σi*) does not correspond to a computation of *C* and by lemma [5.2](#_bookmark6), we know that at least one token has been added to the place *T* . By lemma [5.3](#_bookmark7), one token has been lost from the set of places {*c*1*, c*2*, K*}. So we can conclude that *m*2*i*+1({*c*1*, c*2*, K*}) *< m*2*i*−1( *c*1*, c*2*, K* ).

{ }

* + (1*b*) *m*2*i*(*c*1) *> v*1 and *m*2*i*(*c*2) *v*2, or *m*2*i*(*c*1) *v*1 and *m*2*i*(*c*2) *> v*2. In that case, after firing the sequence *β*3*β*4*β*5, at least one to-

≥ ≥

ken was added to *T* from the places *c*1 or *c*2 and so by lemma [5.3](#_bookmark7),

*m*2*i*+1({*c*1*, c*2*, K*}) *< m*2*i*−1({*c*1*, c*2*, K*}).

So in the two subcases, we conclude that we have *m*2*i*+1({*c*1*, c*2*, K*}) *< m*2*i*−1({*c*1*, c*2*, K*}).

1. *m*2*i*−1(*c*1) /= 0 or *m*2*i*−1(*c*2) /= 0. In that case, we start from a mark- ing *m*2*i*−1 that does not correspond to an initial configuration of the 2CM. We know that it is not possible to add tokens in the set of places

{*c*1*, c*2*, K*} from *m*2*i*−1 to *m*2*i*+1, in fact, we can only move some tokens from *c*1*, c*2*, K* to *T* . After firing *σi*, two cases are possible.

{ }

* + (2*a*) *m*2*i*(*c*1) = *v*1 and *m*2*i*(*c*2) = *v*2. In that case, firing *β*3*β*4*β*5, we reach a marking *m*2*i*+1 to which we can apply case 1 above.
  + (2*b*) *m*2*i*(*c*1) *> v*1 and *m*2*i*(*c*2) *v*2, or *m*2*i*(*c*1) *v*1 and *m*2*i*(*c*2) *> v*2. In that case, after firing the sequence *β*3*β*4*β*5, at least one to- ken was added to *T* from the places *c*1 or *c*2 and so by lemma 3, *m*2*i*+1({*c*1*, c*2*, K*}) *< m*2*i*−1({*c*1*, c*2*, K*}).

≥ ≥

From cases 1 and 2 above, we have that if *s* is not reachable in *C*, at least

one token is lost (at least one token is put in *T* ) when firing *σiβ*3*β*4*β*5*σi*+1*β*3*β*4*β*5 for any *i* ≥ 1. This guarantees, following Lemma [5.3](#_bookmark7), that the number of tokens in {*c*1*, c*2*, K*} will reach zero after a finite amount of time. This means that *C* will not be able to simulate any increment in *C* and will be blocked. We conclude that *σ* cannot be infinite and, then, cannot satisfy the formula

N

*β*3. This contradicts our hypothesis.

Theorem 5.6 *The place boundedness problem is undecidable for* PN+NBA*.*

Proof. Here, we only sketch the proof. Let *C* = *c*1*, c*2*, L,* Instr be a 2CM. Let us show that we can reduce the boundedness problem for *C* to the place boundedness problem for a PN+NBA.

⟨ ⟩

From the widget N*C* corresponding to *C* we construct a PN+NBA N ' as follows. We add the places *p*1 and *p*2 and the transitions *β*1, *β*2, *β*3 and *β*4 as shown in Figure [4](#_bookmark4)(d). Intuitively, while *p*1 contains a token the transitions *β*1 and *β*2 can be fired and move tokens from *c*1 and *c*2 to the capacity place *K*. So *β*1 and *β*2 can be used to reset *c*1 and *c*2 and put back the tokens in *K*. When *β*3*β*4 are fired the control flow token moves from *p*1 to *l*1 passing through *p*2 and one token is added into *K*. So we extend the simulation capacity of the widget by one. This construction allows us to move all the tokens in *c*1*, c*2 to *K* and put the control token into the initial control flow place. If the counters are not set to zero, non-blocking arcs guarantee the lost of at least one token from {*c*1*, c*2*, K*}. Moreover, for each place *li* such that TypeInst(*li*)= incj we add a transition *βli* that moves the control token into *p*1 and moves one token from *K* to *T* if there is some tokens in *K*. We extend *mk* (*k* ≥ 1) to P such that *mk*( *p*1*, p*2 )=0 and we take *m*1 as initial marking. We have that *K* is unbounded iff *C* is unbounded.

{ }

{ }

'

Suppose that *C* is unbounded. Starting from *m*1, the only way to increment the number of tokens in {*c*1*, c*2*, K*} is to mimic *C* until there is no more tokens in and the next operation to mimics is an increment. Then, firing the transitions *β*1 and *β*2, the counters are set to zero moving all the tokens from

*K*

{*c*1*, c*2} to *K* and one new token is generated into *K* by firing *β*3*β*4. This allows us to reach *m*2. Applying this strategy from any *mi* (*i* ≥ 2) allows us to reach *mi*+1 and leads to the construction of an infinite computation where the number of tokens in {*c*1*, c*2*, K*} grows infinitely often. As all the tokens in the set {*c*1*, c*2*, K*} are moved to *K* at the end of the simulation of *C* by firing *β*1 and *β*2, *K* is unbounded in this computation.

If *C* is bounded, there is *k* ∈ N such that starting from *mk*, it is not possible to faithfully simulates *C* and then fire *β*1 without losing tokens in *c*1*, c*2*, K* by moving tokens to *T* with non-blocking arcs. This ensures the boundedness of *K*.

{ }

# Future Works

Recently, several extensions of the Petri net formalism have been proposed for modeling parametric systems, a.o. Transfer nets [[4](#_bookmark14)], Reset nets [[3](#_bookmark13)], Multi- transfer nets [[7](#_bookmark17)], and the extension proposed in this paper. We have defined the extension of this paper in order to model partially non-blocking rendez- vous. The other extensions have been proposed for similar reasons related to modeling issues. Nevertheless, a careful analysis of the expressive power of those different extensions of Petri net has not been done so far. We plan to compare formally the expressive power of those extensions by studying the languages that they are able to define.

# Conclusion

In this paper, we have studied the decidability of five problems for a simple extension of Petri Nets that makes possible the modeling of “partially non- blocking rendez-vous” (necessary to model multi-threaded JAVA programs). The five problems that we have studied are decidable for the basic Petri Net model. We have shown that due to strict monotonicity of the extended model and thanks to general results on well-structured transition systems, the mark- ing coverability and the boundedness problems remain decidable. On the other hand, the three other problems: marking reachability, action-based LTL model-checking and place boundedness become undecidable. Our results are summarized in Table [1](#_bookmark10).

|  |  |  |
| --- | --- | --- |
| Problems | PN | PN + NBA |
| Marking Reachability | √ | × |
| Marking Covering | √ | √ |
| Boundedness | √ | √ |
| Place Boundedness | √ | × |
| Action-based LTL | √ | × |

Table 1 Summary of the decidability/undecidability results.

“undecidable”.

√ stands for “decidable”, and × for

The reader interested in our results may want to look at the following related works. The decidability of the five problems considered in this paper for the Petri net models can be found in: for boundedness, place boundedness

and covering in [[13](#_bookmark23)], for reachability in [[14](#_bookmark24)], and action-based LTL model- checking in [[10](#_bookmark20)]. Several definition of extended Petri nets can be found in [[4](#_bookmark14)] and in [[5](#_bookmark15)]. Undecidability results for the class of transfer nets can be found in [[5](#_bookmark15),[6](#_bookmark16),[8](#_bookmark18)]. In [[15](#_bookmark25)], similar problems are studied in the context of lossy counter machines. For the practical analysis of models that subsume the class of extended Petri Nets studied here, we refer the reader to [[7](#_bookmark17)].

# Acknowledgement

We would like to thank anonymous reviewers for suggesting improvements to the submitted version of this paper and in particular for suggesting the future work on comparing the expressive power of the different extensions of Petri nets proposed in the context of parametric systems verification.

# References

1. P. A. Abdulla, K. Cerans, B. Jonsson, and Y.-K. Tsay. General Decidability Theorems for Infinite-state Systems. In *Proceedings of the 11th Annual Symposium on Logic in Comuter Science (LICS’96)*, pages 313–321. IEEE Computer Society Press, 1996.
2. K.R. Apt and D. Kozen. Limits for Automatic program Verification of Finite-State Concurrent Systems. *Information Processing Letters*, 22(6), 1986.
3. J. Billington. *Extensions to Coloured Petri nets and their applications to Protocols*. PhD thesis, University of Cambridge, 1991.
4. G. Ciardo. Petri nets with marking-dependent arc multiplicity: properties and analysis. In *Proceeding of the 15th International Conference on Applications and Theory of Petri Nets (ICATPN 94)*, volume 815 of *LNCS*, pages 179–198. Springer, 1994.
5. C. Dufourd, A. Finkel, and Ph. Schnoebelen. Reset Nets Between Decidability and Undecidability. In *In Proceedings of the 25th International Colloquium on Automata, Languages, and Programming (ICALP’98)*, volume 1443 of *LNCS*, pages 103–115. Springer, 1998.
6. C. Dufourd, P. Jancar, and P. Schnoebelen. Boundedness of Reset P/T nets. In *Proceedings of the 26th International Colloquium on Automata, Languages, and Programming (ICALP’99)*, volume 1644 of *LNCS*, pages 301–310. Springer, 1999.
7. G. Delzanno, J-F. Raskin, and L. Van Begin. Towards the Automated Verification of Multithreaded Java Programs. In *Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 2002)*, volume 2280 of *LNCS*, pages 173–187. Springer, 2002.
8. C. Dufourd. *R´eseaux de Petri avec reset/transfert : D´ecidabilit´e et ind´ecidabilit´e*. PhD thesis, ENS de Cachan, 1998.
9. J. Esparza, A. Finkel, and R. Mayr. On the Verification of Broadcast Protocols. In *Proceedings of the 14th Annual Symposium on Logic in Computer Science (LICS’99)*, pages 352–359. IEEE Computer Society Press, 1999.
10. J. Esparza. On the Decidabilty of Model Checking for Several mu-calculi and Petri Nets. In *Proceedings of the 19th International Colloquium on Trees in Algebra and Programming*, volume 787 of *LNCS*, pages 115–129, 1994.
11. A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.
12. S. M. German and A. P. Sistla. Reasoning about Systems with Many Processes. *Journal of ACM*, 39(3):675–735, 1992.
13. R. M. Karp and R. E. Miller. Parallel Program Schemata. *Journal of Computer and System Sciences*, 3:147–195, 1969.
14. E.W. Mayr. An algorithm for the general petri net reachability problem. *SIAM Journal of Computing*, 3(13):441–460, 1984.
15. R. Mayr. Undecidable Problems in Unreliable Computations. In *Proceedings of the 4th Latin American Symposium on Theoretical Informatics (LATIN’2000)*, volume 1776 of *LNCS*, pages 377–386. Springer, 2000.
16. N.M. Minsky. *Finite and Infinite Machines*. Englewood Cliffs, N.J., Prentice-Hall, 1967.

# Appendix

Lemma 5.1 *Let γ* = ⟨*loc*1*, v*1*, v*2⟩⟨*loc*2*, v*2*, v*2⟩ *...* ⟨*locr, v*1*, v*2⟩ *be a computation*

1

1

2

2

*r*

*r*

*of the* 2CM *C such that for any i* ∈ {1*,* 2*,..., n*}*, v*1 + *v*2 ≤ *k. Let* N*C be the*

*i*

*i*

PN+NBA *associated to C. The sequence of transitions α*(*γ*) *is firable from*

*the marking mk and firing this sequence of transitions leads to a marking m*' *defined as follows: m*'(*l*) = 1*, for l* = *locr, m*'(*l*') = 0 *for any l*' /= *locr, m*'(*c*1)= *v*1*, m*'(*c*2)= *v*2*, m*'(*K*)= *k* − *v*1 − *v*2*, and m*'(*T* )= 0*.*

*r*

*r*

*r*

*r*

Proof. By induction on the length of the computations of *C*. The basic case (*l* = 1) is obvious. Suppose that the lemma holds for all the computations of size *l < n*.

Let *γ* = *γ*' · ⟨*locn, v*1*, v*2⟩ be a computation of *C* of size *n* where *γ*' =

*n n*

⟨*loc*1*, v*1*, v*2⟩ *...* ⟨*locn*−1*, v*1 *, v*2 ⟩. By induction hypothesis, we have that

1

1

*n*−1

*n*−1

*α*(*γ*') leads to the marking *m*' in N*C* such that *m*'(*li*) = 1 if *li* = *locn*−1,

*m*'(*li*) = 0 for all *li* ∈ *L* \ {*locn*−1}, *m*'(*c*1) = *v*1 , *m*'(*c*2) = *v*2 , *m*'(*K*) =

*n*−1

*n*−1

1

*k* − *v*

— *v*

*n*−1

2

*n*−1

and *m*'(*T* ) = 0. The following cases hold.

1. If Instr(*locn*−1) is of the form *cj* := *cj* + 1; goto *l*', then we have that

*α*(⟨*locn , v*1 *, v*2 ⟩)= *tr* such that *tr* = ⟨*I, O,* ∅⟩ where *I* = {(*locn*−1*,* 1)*,*

1

*n*−1

*n*−1

(*K,* 1)} and *O* = {(*l*'*,* 1)*,* (*cj,* 1)}. By hypothesis we have *m*'(*K*) *>* 0 and we have *m*' →*tr m*'' such that *m*''(*l*')= 1, *m*''(*li*) = 0 for all *li* ∈ *L* \ {*l*'}, *m*''(*c*1)= *v*1, *m*''(*c*2)= *v*2, *m*''(*K*)= *k* − *v*1 − *v*2 and *m*''(*T* )= 0.

*n*

*n*

*n*

*n*

1. If Instr(*locn*−1) is of the form *cj* := *cj* − 1; goto *l*', then we have that

*α*(⟨*locn*−1*, v*1

*n*−1

*, v*

2

*n*−1

⟩)= *tr* such that *tr* = ⟨*I, O,* ∅⟩ where *I* = {(*locn*−1*,* 1)*,*

(*cj,* 1)} and *O* = {(*l*'*,* 1)*,* (*K,* 1)}. As ⟨*locn*−1*, v*1

*n*−1

*, v*

2

*n*−1

⟩ has a successor,

we have *vj >* 0 and *m*' →*tr m*'' such that *m*''(*l*') = 1, *m*''(*li*) = 0 for

*n*−1

all *li* ∈ *L* \ {*l*'}, *m*''(*c*1) = *v*1, *m*''(*c*2) = *v*2, *m*''(*K*) = *k* − *v*1 − *v*2

and

*n n n n*

*m*''(*T* )= 0.

1. If Instr(*locn*−1) is of the form if *cj* = 0 then goto *l*' else goto *l*'', then

*j n*−1

if *v*

= 0 we have *α*(⟨*locn*−1*, v*1

2

*n*−1

*, v*

⟩) = *tr*=0 such that *tr*=0 =

⟨*I*=0*, O*=0*,* {⟨*cj,T* ⟩}⟩ where *I*=0 = {(*locn*−1*,* 1)} and *O*=0 = {(*l*'*,* 1)}. *tr*=0

*n*−1

is firable from *m*' and we have *m*' →*tr m*'' such that *m*''(*l*')= 1, *m*''(*li*)= 0 for all *li* ∈ *L* \ {*l*'}, *m*''(*c*1)= *v*1, *m*''(*c*2)= *v*2, *m*''(*K*)= *k* − *v*1 − *v*2 and

*n*

*n*

*n*

*n*

*m*''(*T* )= 0. Otherwise if *vj*

*n*−1

*n*−1

*, v*

*>* 0 we have *α*(⟨*locn*−1*, v*1

2

*n*−1

⟩)= *tr*/=0

such that *tr*/=0 = ⟨*I*/=0*, O*/=0*,* {⟨*cj,T* ⟩}⟩ where *I*/=0 = {(*locn*−1*,* 1)*,* (*cj,* 1)}

and *O*/=0 = {(*l*''*,* 1)*,* (*cj,* 1)}. *tr*/=0 is firable from *m*' and we have *m*' →*tr*

*m*'' such that *m*''(*l*'') = 1, *m*''(*li*) = 0 for all *li* ∈ *L* \ {*l*''}, *m*''(*c*1) = *v*1,

*n*

*m*''(*c*2)= *v*2, *m*''(*K*)= *k* − *v*1 − *v*2 and *m*''(*T* )= 0.

*n n n*

Lemma 5.2 *Let σ* = *tr*1*tr*2 *... trn be a sequence of transitions of the* PN+NBA

*C associated to the* 2CM *C. If mk σ m*' *and m*'(*T* ) = 0*, then α*−1(*σ*) *is a computation of C such that* final(*α*−1(*σ*)) = *loc, v*1*, v*2) *where loc* = *l if m*'(*l*)= 1*, v*1 = *m*'(*c*1) *and v*2 = *l*'(*c*2)*.*

⟨ ⟩

N →

Proof. By induction on the size of the sequence of transitions. The basic case (*l* = 1) is obvious. Suppose that the lemma holds for all the sequences of transitions of size *l < n*. Let *σ* = *σ*' *trn* be a sequence of transitions of

·

'

N*C* of size *n* where *σ* = *tr*1 *... trn*−1. By induction hypothesis we have that

*m* →*σ*' *m*' and final(*α*−1(*σ*')) = ⟨*loc, v*1*, v*2⟩ such that *m*'(*loc*)= 1, *m*'(*c* )= *v*1,

1

1

1

*m*'(*c*2)= *v*2 and *m*'(*T* ) = 0. The following cases holds.

1. if Instr(*α*−1(*trn*)) is of the form *cj* := *cj* + 1; goto *l*', then *trn* = ⟨*I, O,* ∅⟩ such that *I* = {*loc, K*} and *O* = {*l*'*, cj*}. We have *m*' →*trn m*'' and *α*−1(*σ*) is a computation of *C* with final(*α*−1(*σ*)) = ⟨*l*'*, v*1*, v*2⟩ such that

2 2

*m*''(*l*')= 1, *m*''(*c*1)= *v*1, *m*''(*c*2)= *v*2 and *m*''(*T* )= 0.

2 2

1. if Instr(*α*−1(*trn*)) is of the form *cj* := *cj* − 1; goto *l*', then *trn* = ⟨*I, O,* ∅⟩

such that *I* = {(*loc,* 1)*,* (*cj,* 1)} and *O* = {(*l*'*,* 1)*,* (*K,* 1)}. We have *m*' →*trn*

*m*'' and *α*−1(*σ*) is a computation of *C* with final(*α*−1(*σ*)) = ⟨*l*'*, v*1*, v*2⟩ such

2 2

that *m*''(*l*')= 1, *m*''(*c*1)= *v*1, *m*''(*c*2)= *v*2 and *m*''(*T* )= 0.

2 2

1. if Instr(*α*−1(*trn*)) is of the form if *cj* = 0 then goto *l*' else goto *l*'', then if *m*'(*cj*) = 0, *trn* must be such that *trn* = ⟨*I, O,* {⟨*cj,T* ⟩}⟩ with *I* =

{(*loc,* 1)} and *O* = {(*l*'*,* 1)}. We have *m*' →*trn m*'' and *α*−1(*σ*) is a computation of *C* with final(*α*−1(*σ*)) = ⟨*l*'*, v*1*, v*2⟩ such that *m*''(*l*') = 1,

2

2

*m*''(*c*1) = *v*1, *m*''(*c*2) = *v*2 and *m*''(*T* ) = 0. Otherwise if *m*'(*cj*) *>* 0,

2 2

*tr* must be such that *tr* = ⟨*I, O,* ∅⟩ with *I* = {(*loc,* 1)*,* (*cj,* 1)} and *O* =

{(*l ,* 1)*,* (*cj,* 1)}, otherwise *T* would contain one token after firing *trn*. We have *m*' →*trn m*'' and *α*−1(*σ*) is a computation of *C* with final(*α*−1(*σ*)) =

''

⟨*l*''*, v*1*, v*2⟩ such that *m*''(*l*'')= 1, *m*''(*c*1)= *v*1, *m*''(*c*2)= *v*2 and *m*''(*T* )=

2 2 2 2

0.

Lemma 5.3 *Let* N*C be the* PN+NBA *associated to the* 2CM *C. For any marking m* ∈ Reach(N*C, mk*)*, we have that m*(*c*1*, c*2*, K,T* )= *k.*

Proof. By induction on the size of the minimal computation of *C* that allows us to reach *m*. The basic case (*l* = 1) is obvious. Suppose that the lemma holds for all the markings reachable in *i* steps from *mk* in N*C* with *i < n*. Suppose that *m* is reachable by firing *n* − 1 transitions and we have *m* →*tr m*' for some transition *tr* of N . *tr* can be of the following forms:

N

1. *tr* = ⟨*I, O,* ∅⟩ with *I* = {(*l,* 1)*,* (*cj,* 1)} and *O* = {(*l*'*,* 1)*,* (*K,* 1)} and corresponds to a decrement. In this case we have *m*({*c*1*, c*2*, K,T* }) = *m*'({*c*1*, c*2*, K,T* }).
2. *tr* = ⟨*I, O,* ∅⟩ with *I* = {(*l,* 1)*,* (*K,* 1)} and *O* = {(*l*'*,* 1)*,* (*cj,* 1)} and corresponds to an increment. In this case we have *m*({*c*1*, c*2*, K,T* }) = *m*'({*c*1*, c*2*, K,T* }).
3. *tr* = ⟨*I, O,* {⟨*cj,T* ⟩}⟩ with *I* = {(*l,* 1)} and *O* = {(*l*''*,* 1)} and corresponds to a test for zero on *cj*. In this case, when *m*(*cj*) = 0 or *m*(*cj*) *>* 0, we have *m*({*c*1*, c*2*, K,T* })= *m*'({*c*1*, c*2*, K,T* }).
4. *tr* = ⟨*I, O,* ∅⟩ with *I* = {(*l,* 1)*,* (*cj,* 1)} and *O* = {(*l*''*,* 1)*,* (*cj,* 1)} and corresponds to a test for zero on *cj* when *m*(*cj*) *>* 0. In this case we have

*m*({*c*1*, c*2*, K,T* })= *m*'({*c*1*, c*2*, K,T* }).