

Electronic Notes in Theoretical Computer Science 223 (2008) 239–264

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

Reachability in Petri Nets with Inhibitor Arcs

Klaus Reinhardt[1](#_bookmark0)

*Wilhelm-Schickhard Institut fu¨r Informatik Universit¨at Tu¨bingen*

*Sand 13, D-72076 Tu¨bingen, Gemany*

**Abstract**

We define 2 operators on relations over natural numbers such that they generalize the operators ’+’ and ’\*’ and show that the membership and emptiness problem of relations constructed from finite relations with these operators and *∪* is decidable. This generalizes Presburger arithmetics and allows to decide the reachability problem for those Petri nets where inhibitor arcs occur only in some restricted way. Especially the reachability problem is decidable for Petri nets with only one inhibitor arc, which solves an open problem in [[11](#_bookmark41)]. Furthermore we describe the corresponding automaton having a decidable emptiness problem.

*Keywords:* Petri nets, inhibitor arcs, reachability, monotone transitive closure, counter automata

# Introduction

The decidability of the reachability problem in Petri nets without inhibitor arcs is proved in [[14](#_bookmark44)] and later in [[12](#_bookmark42)] and [[13](#_bookmark43)]. On the other hand, the reachability problem is undecidable for Petri nets with two inhibitor arcs which follows from [[15](#_bookmark45)]. An open problem in [[11](#_bookmark41)] was the reachability problem for Petri nets with one inhibitor arc.

An important method is the use of semilinear sets which are defined using the operators +, *∗*, *∪* over finite sets of vectors (multisets). Semilinear sets are the solutions of Presburger formula, where Presburger arithmetic is the first order logic over the natural numbers and the addition. Presburger arithmetic is decidable and semilinear sets are closed under *∩* and complement [[8](#_bookmark35)],[[5](#_bookmark36)].

But a reachability relation for a Petri net is in general not semilinear. For that reason the basic idea of this paper is to replace + and *∗* by suitable operations *◦Q* and *∗Q* which are able to express a reachability relation as the sequence of relations

(like the transitive closure used in [[9](#_bookmark37)] to characterize NL with first order logic or more generally in [[1](#_bookmark31)]).

1 Email: [reinhard@informatik.uni-tuebingen.de](mailto:reinhard@informatik.uni-tuebingen.de)

1571-0661 © 2008 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.2008.12.042

But the transitive closure over first order logic over natural numbers with the addition immediately becomes undecidable. For that reason the important principle of monotonicity in the reachability relation of Petri nets is combined with the idea of the transitive closure. That means the operator *∗Q* is a monotone transitive closure (see Corollary [3.6](#_bookmark15)). We consider the following three steps:

* 1. One application of *∗Q* already allows us to express the reachability problem in a Petri net without inhibitor arcs (Lemma [2.1](#_bookmark5)).
  2. A second application of *∗Q* (containing the first one in a nested way) allows us to express the reachability problem in a Petri net with one inhibitor arc (Lemma [2.4](#_bookmark9)).
  3. Arbitrary nested applications of *∗Q* allow us to express the reachability problem in a Petri net for which there exists an ordering of the places such that a place has an inhibitor arc to all those transitions which have an inhibitor arc from a preceding place (Theorem [5.1](#_bookmark27)).

In Section [3](#_bookmark10) we use expressions consisting of the operators *∪*, *◦Q* and *∗Q* on sets of multisets in a special form (Lemmata [3.1](#_bookmark11) and [3.5](#_bookmark14) show that we can bring every such expression in this form), which models the idea of a nested Petri net: The firing behavior of a complex (nested) transition is linked to firing sequences in inner Petri nets by a semilinear relation (unlike in the structured nets in [[2](#_bookmark32)]). The connection between these inner Petri nets corresponds to the chain of vector addition systems used in [[12](#_bookmark42)] and it is described by the same semilinear relation. The main difference to the structure of the proofs in [[12](#_bookmark42)] and [[13](#_bookmark43)] is that states are not anymore necessary since their function is instead fulfilled (Section [4.4](#_bookmark22)) by the nestedness of expressions (like regular expressions replace a finite automaton).

Furthermore we define a condition (normal form *T* corresponding to the property Θ in [[12](#_bookmark42)]), which allows to check the emptiness of the expressed set of multisets, we define a size of the expressions leading to a Noetherian order and construct an algorithm in Section [4](#_bookmark16) which finds an equivalent expression fulfilling condition *T* . Each step of the algorithm constructs an equivalent expression which is smaller with respect to the defined size. Some kind of pumping property (Lemma [4.3](#_bookmark24)) ensures the existence of arbitrary high firing sequences if condition *T* is fulfilled.

The most general result (see Step 3 above) is described in Section [5](#_bookmark26) This allows to decide the expressed reachability problem. Sections [6](#_bookmark28) and [7](#_bookmark30) describe the conclusions for emptiness problems for automata.

The contents of this paper mainly corresponds to Chapter 5 in the habilitation- thesis [[17](#_bookmark47)], which also contains the proofs which had to be left out here.

* 1. *Multisets*

For the sake of a flexible description, we use multi-sets instead of vectors. A *multi- set* over *B* is a function in N*B*.

We might write a multiset **f** *∈* N*B* as a set *{b '→ f* (*b*) *| b ∈ B}*, as a table

*b*1 *, b*2

*, ..., bn*

0 **f** (*b*1)1

or as an *n*-ary vector B C. For the latter, we have to

**f** (*b*2)

CA

B .

*f* (*b*1) *f* (*b*2)

*f* (*bn*)

.

.

@

**f** (*bn*)

assume an ordering on *B* = *{b*1*, b*2*, ..., bn}* (without relevance to the contents), and

in the first two descriptions, we only need to write those *b*’s with *f* (*b*) *>* 0. Although we do not a priori limit the size of *B*, we only use multisets for a finite *B* in this paper. For multisets, we use the variables **c***,* **d***,* **e***,* **f** *,* **g***,* **h***,* **m***,* **n***,* **r***,* **s***,* **x***,* **y**, and for sets of multisets, we use the capitals **E***,* **L***,* **M***,* **N***,* **R** and **I***d* (the latter will denote the

identity for the operator *◦Q* to be defined).

For *A ⊆ B*, we regard functions in N*A ⊆* N*B* as extended to zero for undefined values. This allows us to add any two multisets **f** *∈* N*A* and **g** *∈* N*B* and obtain a multiset in (**f** + **g**) *∈* N*A∪B* with (**f** + **g**)(*x*) = **f** (*x*) + **g**(*x*) in the same way as we would add the corresponding vectors assuming an ordering on *A ∪ B*. The neutral element for addition is *∅* with *∅*(*x*) = 0 for all *x*. It holds N*A ∩* N*B* = N*A∩B*.

The restriction **f** *|A* of a multi-set **f** *∈* N*B* to *A* is

**f** *|A* (*b*) := **f** (*b*) if *b ∈ A* else **f** *|A* (*b*) := 0*.*

This means **f** *|A*:= *{b '→* **f** (*b*) *| b ∈ A}* . The complement operator is **f** *|A*:= *{b '→*

**f** (*b*) *| b /∈ A}* , thus **f** = **f** *|A* +**f** *|A*.

For a finite set **M** = *{***m**1*, ...,* **m***k}⊆* N*A* of multi-sets,

**M***∗* := *{ a*1**m**1 + *...* + *ak***m***k| ∀i ≤ k ai ∈* N*}*

is the set of all linear combinations generated by **M**. More generally, by **M**0 := *{∅}*

and **M***i*+1 := **M***i* + **M**, we can define **M***∗* := *i* **M***i*.

* + 1. *New operator on multisets*

For an unambiguous [2](#_bookmark1) and injective binary relation *Q*, we define the operator *◦Q*

on two sets of Multisets **M** and **N** as

,

**N***◦Q***M** := **n** *|π*1(*Q*) +**m** *|π*2(*Q*) **n** *∈* **N***,* **m** *∈* **M***, ∀*(*a, b*) *∈ Q* **n**(*a*) = **m**(*b*) *.*

This means if **n** and **m** “match” according to *Q*, then the values for an *a ∈ π*1(*Q*) =

*{a|*(*a, b*) *∈ Q}* in **n** and the values for a *b ∈ π*2(*Q*) = *{b|*(*a, b*) *∈ Q}* in **m** are “used up against each other” and the rest is added. For example,

0 1 0 1

0 1 0 1 0 1

0 1 0 1 0 1

3 2

B@6CA*,* B@5CA

* *​{*(*b*1*,b*2)*}*

8 7 5

B@3CA*,* B@2CA*,* B@2CA =

8 7 5

B@6CA*,* B@5CA*,* B@5CA

1 2 1 2 3

or

1. 4 5

031 021 081 071 051

11! 9!

B@6CA*,* B@5CA

1

2

3

1

2

* *​{*(*b*3*,b*3)*}*

B@3CA*,* B@2CA*,* B@2CA = 9 *,* 7

The latter example shows that the dimension is necessarily reduced (*b*3 is used up on both sides) if *π*1(*Q*)*∩π*2(*Q*) is not empty. We will later need *◦*ˆ*A* := *◦{*(*a,a*)*|a∈A}*

2 A binary *Q* is unambiguous if *Q−*1 is injective.

to get **N***◦*ˆ*A***M** = **n** *|A* +**m** *|A* **n** *∈* **N***,* **m** *∈* **M***, ∀a ∈ A* **n**(*a*) = **m**(*a*) . For example

  }

* *{*(*b*3*,b*3)*}* = *◦*ˆ*{b*3*}*.

If *π*1(*Q*) and *π*2(*Q*) are disjoint, we define **I***dQ* := *{{a '→* 1*,b '→* 1*}|* (*a, b*) *∈ Q}∗* which is the neutral element for *◦Q*. Obviously, it holds **N***◦∅***M** = **N** + **M** which makes + with the neutral element **I***d∅* = *{∅}* a special case of the *◦Q* operator.

Furthermore, for *Q* with *π*1(*Q*) and *π*2(*Q*) disjoint, we define *∗*0 (**M**) := **I***dQ*,

*Q*

*∗i*+1(**M**) := *∗i* (**M**)*◦Q*(**M** + **I***dQ*) and *∗Q*(**M**) := *∗i* (**M**). Again, *∗∅*(**M**) = **M***∗*

*Q*

*Q*

*i*

*Q*

is a special case. It is easy to see that *∗Q*(**M**) is the closure of **M** *∪* **I***dQ* under *◦Q*

and the addition *◦∅*:

If **f** *,* **g** *∈ ∗Q*(**M**) then there are *i, j ∈* N with **f** *∈ ∗i* (**M**) and **g** *∈ ∗j* (**M**) thus

*Q Q*

*{***f** *}◦Q{***g***}⊆ ∗i* (**M**)*◦Q∗j* (**M**) = *∗i*+*j* (**M**) *⊆ ∗Q*(**M**).

*Q Q Q*

Let **f** *'* = **f** + *{a '→* **g**(*b*)*,b '→* **g**(*b*) *|* (*a, b*) *∈ Q} ∈* **f** + **I***dQ* and **g***'* = **g** +

+**g** *|*

*{a '→* **f** (*a*)*,b '→* **f** (*a*) *|* (*a, b*) *∈ Q} ∈* **g** + **I***dQ*. Then **f** + **g** = **f** *' |*

*π*1(*Q*)

*'*

*π*2(*Q*)*∈*

*{***f** *'}◦Q{***g***'} ⊆ ∗i* (**M**)*◦Q∗j* (**M**) = *∗i*+*j* (**M**) *⊆ ∗Q*(**M**) according to the definition

*Q Q Q*

of *◦Q*. Clearly *∗Q*(**M**) = *∗Q*(**M**)+ **I***dQ*.

For example, for *i >* 0 we have

*∗*  021 = 0

*i*

*{*(*b*1*,b*2)*}*

B@5CA

B@2+ 3*i* + *j* CA *j ≥*

021

2+ *k*

1

2+ *j* 1

0 leading to

1

0

1

*∗i*+1

*{*(*b*1*,b*2)*}*

B@5CA

1

= *∗i*

B@5CA

* + *{*(*b ,b* )*}*

B@5+ *k* CA *k ≥* 0

1

= B@5+ 3*i* + *j* CA *j ≥* 0

*i* +1

*i* 0

2

1

1

1

2

(by “matching” with *k* = 3*i* + *j*) and

*∗*  021 = 0 *j* 1 0

*{*(*b*1*,b*2)*}*

0 2+ *j*

2+ *j* 1 0

*{*(*b*1*,b*2)*}*

B@5CA

1

B@ *j* CA*,* B@2+ 3*i* + *j* CA *i, j ≥*

0

*i*

**Remark:** Adding **I***dQ* is a crucial point: It corresponds to the monotonicity in Petri nets. Without this, deciding emptiness for the expressions would become undecidable.

* 1. *Properties of the new operators*

Obviously, it holds **N***◦Q***M** = **M***◦Q−*1 **N**. Furthermore, we can express the inter- section of **N***,* **M** *⊆* N*A* by **N***◦Q'* **L***◦Q''* **M** = **N** *∩* **M** with *Q'* := *{*(*a, a'*) *| a ∈ A}*, *Q''* := *{*(*a'', a*) *| a ∈ A}* and **L** := *{{a '→* 1*, a' '→* 1*, a'' '→* 1*}| a ∈ A}∗*.

Note here that, in general, **N***◦Q'* **L***◦Q''* **M** can only be written without brackets because *π*1(*Q''*)*∪*(*{a | ∃* **f** *∈* **M***,* **f** (*a*) *>* 0*}\π*2(*Q''*)) and *π*2(*Q'*)*∪*(*{a | ∃* **f** *∈* **N***,* **f** (*a*) *>* 0*}\ π*1(*Q'*)) are disjoint. If, additionally, *π*2(*Q''*) and *{a | ∃* **f** *∈* **N***,* **f** (*a*) *>* 0*}* are disjoint and *{a | ∃* **f** *∈* **M***,* **f** (*a*) *>* 0*}* and *π*1(*Q'*)) are disjoint, then **N***◦Q'* **L***◦Q''* **M** = **L***◦Q'−*1*∪Q''* (**M** + **N**).

* 1. *Semilinearity*

The class of semilinear sets is the smallest class of sets of multisets containing all finite sets of multisets and being closed under *∪,* + and *∗*. The semilinear sets are also closed under *∩*, as shown in [[8](#_bookmark35)] and [[5](#_bookmark36)]. As a normal form to express semilinear sets, we will use the union of linear sets of the form **c** + Γ*∗* for a constant **c** *∈* N*A* and periods Γ *⊆* N*A*.

The operator *◦Q* preserves semilinearity: Assume **N** and **M** are semilinear sets over *A*, then

**N***'* := *{***f** *' | ∃***f** *∈* **N***∀a ∈ π*1(*Q*) **f** *'*(*a'*) = **f** (*a*) *∧* **f** *'*(*a*) = 0 *∧ ∀a ∈/ π*1(*Q*) **f** *'*(*a*) = **f** (*a*)*},*

**M***'* := *{***f** *' | ∃***f** *∈* **M***∀a ∈ π*2(*Q*) **f** *'*(*a'*) = **f** (*a*) *∧* **f** *'*(*a*) = 0 *∧ ∀a ∈/ π*2(*Q*) **f** *'*(*a*) = **f** (*a*)*},*

*'* := *{{a' '→* 1*, b' '→* 1*}, {c '→* 1*}|* (*a, b*) *∈ Q, c ∈ A}∗*

**E**

*Q*

= *{***f** *| ∀*(*a, b*) *∈ Q* **f** (*a'*) = **f** (*b'*)*}*

are as well semilinear sets over the set *A ∪ π*1(*Q*)*' ∪ π*1(*Q*)*'* which is extended by

new elements. Thus, **N***◦Q***M** = ((**N***'* + **M***'*) *∩* **E***'* ) *| ' '* is semilinear. Since

*Q π*1(*Q*) *∪π*1(*Q*)

the closure under *∩* is effective, this will allow a decision algorithm to continue with

the representation

*l*

*Lj*

*j*=1

by linear sets

*Lj*.

On the other hand, *∗Q* does not preserve semilinearity:

011

001*∗*

0*a*1

Let **M** := B@0CA + B@1CA , then *∗{*(*b ,b* )*}*(**M**) = B@ *b*CA *c ≤ b*2*a*

is not semilinear.

3 2

0 2 *c*

# The reachability relation for Petri nets

* 1. *The reachability relation for Petri nets without inhibitor arcs*

We describe a *Petri net* as the triple *N* = (*P, T, W* ) with the places *P* , the transitions *T* and the weight function *W ∈* N*P ×T ∪T ×P* . A transition *t ∈ T* can fire from a marking **m** *∈* N*P* to a marking **m***' ∈* N*P* , denoted by **m**[*t⟩***m***'*, if

**m** *− W* (*., t*) = **m***' − W* (*t, .*) *∈* N*P .*

A *ﬁring sequence w* = *t*1*...tn ∈ T∗* can fire from **m**0 to **m***n*, denoted by **m**0[*w⟩***m***n*, if **m**1*, ...***m***n−*1 exist with **m**0[*t*1*⟩***m**1[*t*2*⟩...*[*tn⟩***m***n*. The *reachability problem* is to decide for a given net *N* with start- and end markings **m**0*,* **m***e ∈* N*P* , if there is a *w ∈ T∗* with **m**0[*w⟩***m***e*.

Let *P* + := *{p*+ *| p ∈ P}* and *P−* := *{p− | p ∈ P}* be copies of the places. For any multiset, **m** we define the corresponding copies **m***−* := *{p− '→* **m**(*p*) *| p ∈ P}* and **m**+ := *{p*+ *'→* **m**(*p*) *| p ∈ P}*. Then, we can define the *reachability relation* for

a transition *t* as

**R**(*t*) := **m***−* + **m***'*+ **m**[*t⟩***m***'*,

= **r** *∈* N*P* +*∪P− ∀p ∈ P* **r**(*p−*) *− W* (*p, t*) = **r**(*p*+) *− W* (*t, p*) *∈* N,

and the reachability relation for a set of transitions *T* as **R**(*T* ) := **R**(*t*).

*t∈T*

The important property of monotonicity means that whenever **m**[*w⟩***m***'*, then

also (**m**+**n**)[*w⟩*(**m***'*+**n**) for any **n** *∈* N*P* . This corresponds to adding **I***dP* := **I***dP*ˆ with *P*ˆ := *{*(*p*+*, p−*) *| p ∈ P}* and **R**(*t*) can be written as the linear set **R**(*t*) = **c***t* + **I***dP* using **c***t* with **c***t*(*p−*) := *W* (*p, t*) and **c***t*(*p*+) := *W* (*t, p*) for all *p ∈ P* . The reachability relation for the concatenation of two firing sequences is described by **R**(*w*1*w*2) =

**R**(*w*1)*◦P* **R**(*w*2) with *◦P* := *◦P*ˆ and the iteration is done by *∗P* := *∗P*ˆ. We define the reachability relation of the Petri net *N* as **R**(*N* ) := **R**(*T∗*) := *∗P* (**R**(*T* )). The

reachability problem formulates as (**m***−* + **m**+) *∈* **R**(*N* ).

0 *e*

**Lemma 2.1** *There is a ﬁring sequence w ∈ T∗ with* **m**0[*w⟩***m***e in N if and only if*

**m**+*◦P* **R**(*N* )*◦P* **m***−* = (**m***−* + **m**+)*◦*ˆ*P* +*∪P−* **R**(*N* ) = *{∅}.*

0 *e* 0 *e*

*(If there is no w ∈ T∗ with* **m**0[*w⟩***m***e, we have* (**m***−* + **m**+)*◦*ˆ*P* +*∪P−* **R**(*N* ) = *∅.)*

0 *e*

**Proof.** There is a firing sequence *t*1*...tn ∈ T∗* with **m**0[*t*1*⟩***m**1[*t*2*⟩...*[*tn⟩***m***n* = **m***e* in

*N* if and only if (**m***−* + **m**+) *∈* **R**(*ti*) for all 1 *≤ i ≤ n* if and only if (**m***−* + **m**+) *∈*

*i−*1 *i* 0 *e*

**R**(*t*1)*◦P* **R**(*t*2)*...◦P* **R**(*tn*) if and only if (**m***−* + **m**+) *∈* **R**(*T* )*◦P* **R**(*T* )*...◦P* **R**(*T* ) =

0 *e*

*∗n* (**R**(*T* )) if and only if (**m***−* + **m**+) *∈ ∗P* (**R**(*T* )) = **R**(*N* ) and only this multiset

*P* ˆ0 *e*

can “match” according to *P* .

## Example:

Consider the following Petri net *N* with

**R**(*t*1) = *{p− '→* 1*, p*+ *'→* 2*}* + **I***dP* , ***,,***

2 1 *t*1 ***)*** *p*1

**R**(*t*2) = *{p− '→* 3*, p*+ *'→* 2*}* + **I***dP* , 2 ***J***

1 2 *−* + *−* + , ***^***

and thus, **R**(*T* ) =

*p*2 *, p*1

*,*

*p*1 *, p*2

.

3

3

1

2

1

***7***

1

2

0 = *{p*1 *'→*

*t*2

Let furthermore **m**

***,,***

and **m***e* = *{p*1 *'→* 1*, p*2 *'→* 2*}*.

*, p*2 *'→ }*

*p*2 ***(***2

***J***

By concatenating, we get for example

**R**(*t t* ) = **R**(*t* )*◦* **R**(*t* ) = *p− , p− , p*+ + **I***d*

1 2

1

*P*

2

2

1

1

1

2

2

*P*

and

*p*

***^***

2

**R**(*t t t* ) = *p− , p*+ *, p*+ + **I***d* . 2 ***)***

1 1 2

2

2

2

2

1

1

*P*

***< <***

In this way we get (**m***−* + **m**+) = *p− , p− , p*+ *, p*+

***z***

) = *p ,*

*p* + **I**

0

*e*

1

1

2

1

1

1

2

2

1

***z***

)

`

*∈* **R**(

*,*

*p*

1 *,*

***z*** *t*2

*p*

*t*1*t*2*t*1*t*1*t*2

*− −* + +

1 2 2

1 1 1 2

*dP ⊆*

**R**(*N* ).

*t*1 ***z***

***za*** ) ***)***

1 2 3 *p*1

By further concatenating and iterating, we get

**R**((*t t* )*∗*) = *p− , p*+ + *p− , p*+ *∗* + **I***d ,*

1 2

2

1

2

1

1

1

2

1

*P*

**R**(*t t* ) = **R**(*t* )*◦* **R**(*t* ) = *p− , p*+ *, p*+ + **I***d ,*

2 1

2

*P*

1

1

3

2

1

1

2

*P*

**R**((*t t* )*∗*) = *p− , p*+ + *p− , p*+ *∗* + **I***d ,*

2 1

1

2

1

2

1

1

2

1

*P*

**R**(*t t t* ) = *p− , p*+ + **I***d ,*

2 1 1

1

3

1

4

*P*

**R**((*t t t* )*∗*) = *p− , p*+ + *p*+ *∗* + **I***d ,*

2 1 1

1

3

1

3

1

1

*P*

**R**((*t t t* )*∗*) = *p− , p*+ + *p*+ *∗* + **I***d ,*

1 1 2

2

2

2

2

1

1

*P*

**R**((*t t t* )*∗*) = *p− , p*+ *, p− , p*+ + *p*+ *∗* + **I***d ,*

*...,*

1 2 1

2 2 1 2 1

1 1 1 1 1

*P*

which finally yields **R**(*N* ) = **R**(*T∗*) = *∗*

*p− , p*+ *,* *p− , p*+ , =

1 2

2

1

*P*

1 2 3 2

*p− , p*+ *,* *p− , p*+ ,*∗* + **I***d ∪*

2

1

1

2

1

3

2

2

*P*

*p− , p*+ *,* *p− , p*+ , + *p− , p*+ *,* *p− , p*+ *,* *p− , p*+ ,*∗* + **I***d ∪*

2

1

2

1

1

2

1

2

1

1

2

1

2

1

1

2

1

3

2

2

*P*

*p− , p*+ *,* *p− , p*+ *,* *p− , p*+ *, p− , p*+ *,* *p− , p*+ *,* *p− , p*+ *, p*+ *,* *p− , p− , p*+ *,*

1

3

1

3

2

2

2

2

2

1

2

1

1

1

2

1

1

3

2

2

2

2

1

2

2

1

1

1

2

1

1

3

*p− , p*+ , + *p− , p*+ *,* *p− , p*+ *,* *p− , p*+ *,* *p*+ *,* *p*+ ,*∗* + **I***d*

2

2

1

4

2

1

1

2

1

3

2

2

1

1

2

1

1

1

2

1

*P*

* 1. *Petri nets with inhibitor arcs*

An *inhibitor arc* from a place to a transition means that the transition can only fire if no token is on that place. We describe such a Petri net as the 6-tuple (*P, T, W, I,* **m**0*,* **m***e*) with the places *P* , the transitions *T* , the weight function *W ∈* N*P ×T ∪T ×P* , the inhibitor arcs *I ⊆ P ×T* and, the start and end markings **m**0*,* **m***e ∈* N*P* . We will denote an inhibitor arc in the pictures by ————*•* .

A transition *t ∈ T* can fire from a marking **m** *∈* N*P* to a marking **m***' ∈* N*P* , denoted by **m**[*t⟩***m***'* if

**m** *− W* (*., t*) = **m***' − W* (*t, .*) *∈* N*P* and *∀p ∈ P* (*p, t*) *∈ I →* **m**(*p*) = 0*.*

A firing sequence *w* = *t*1*...tn ∈ T∗* can fire from **m**0 to **m***n*, denoted by **m**0[*w⟩***m***n*, if there exist intermediate markings **m**1*, ...***m***n−*1 with **m**0[*t*1*⟩***m**1[*t*2*⟩...*[*tn⟩***m***n*.

The *reachability problem* for a Petri net (*P, T, W, I,* **m**0*,* **m***e*) is to decide, whether there exists a *w ∈ T∗* with **m**0[*w⟩***m***e*.

In the following two lemmata, we restrict the cases for which we have to regard the reachability problem. (See the [[17](#_bookmark47)] for proofs).

**Lemma 2.2** *Each Petri net* (*P, T, W, I,* **m**0*,* **m***e*) *can be changed in such a way that the condition ∀p ∈ P, t ∈ T* (*p, t*) *∈ I → W* (*t, p*) = 0 *holds without changing the inhibitor arcs or the reachability problem.*

**Lemma 2.3** *Each Petri net* (*P, T, W, I,* **m**0*,* **m***e*) *can be changed in a way such that the condition ∀p ∈ P, t ∈ T* (*p, t*) *∈ I →* **m**0(*p*) = **m***e*(*p*) = 0 *holds by changing neither the inhibitor arcs, the condition in Lemma* [*2.2*](#_bookmark6) *nor the reachability problem.*

* 1. *The reachability relation for Petri nets with one inhibitor arc*

Let us consider a Petri-net *N* = (*P, T, W, {*(*p*1*, t*ˆ)*},* **m**0*,* **m***e*) having the property of lemmata [2.2](#_bookmark6) and [2.3](#_bookmark7). As in the case of no inhibitor arcs, we can describe by

**R**(*N'*) = *∗P* (**R**(*T \ {t*ˆ*}*)) for the Petri net *N'* = (*P, T \ {t*ˆ*},W |* ˆ ˆ ) the

*P×{t}∪{t}×P*

reachability relation for firing sequences *w ∈* (*T \ {t*ˆ*}*)*∗*. The restriction to those firing sequences starting and ending with markings without tokens on *p*1 can be described by **R**(*tp* (*N'*)) := *{***r** *∈* **R**(*N'*) *|* **r**(*p−*) = **r**(*p*+) = 0*}*. In **R**(*N* ) :=

1 1 1

*∗P\{p }*(**R**(*tp* (*N'*)) *∪* **R**(*t*ˆ)) with **R**(*t*ˆ) = **c**ˆ+ **I***dP\{p }* we iterate these parts together

1 1 *t* 1

with the alternative of using *t*ˆ. Generalizing Lemma [2.1](#_bookmark5) we get the following:

**Lemma 2.4** *Given a Petri-net N* = (*P, T, W, {*(*p*1*, t*ˆ)*},* **m**0*,* **m***e*) *with only one in- hibitor arc* (*p*1*, t*ˆ) *having the property of lemmata* [*2.2*](#_bookmark6) *and* [*2.3*](#_bookmark7)*, then there is a ﬁring sequence w ∈ T ∗ with* **m**0[*w⟩***m***e if and only if*

**m**+*◦P\{p }***R**(*N* )*◦P\{p }***m***−* = (**m***−* + **m**+)*◦*ˆ + *−*

+ *−* **R**(*N* ) = *{∅}.*

0 1 1 *e*

0 *e P ∪P*

*\{p*1 *,p*1 *}*

*(If there is no such w ∈ T∗, we have* (**m***−* + **m**+)*◦*ˆ + *−*

+ *−* **R**(*N* ) = *∅.)*

0 *e P ∪P*

*\{p*1 *,p*1 *}*

**Proof.** Since **m**0(*p*) = **m***e*(*p*) = 0 according to Lemma [2.3](#_bookmark7), a firing sequence *w ∈ T∗* can be decomposed in minimal firing sequences *w*1*...wk* = *w* having the property **m**0[*w*1*⟩***m**1[*w*2*⟩...*[*wk⟩***m***k* with **m***k* = **m***e* such that **m***i*(*p*1) = 0 for all *i ≤ k*.

Each *wi* is either in (*T \ {t*ˆ*}*)*∗* or equal to *t*ˆ by Lemma [2.2](#_bookmark6) and minimality of *wi*. This holds since the occurrence of *t*ˆ in a *wi* with *|wi| >* 1 would mean that, at that time during the firing of *wi*, there is no token on *p*1, and thus, *wi* would not be minimal.

*i*

If *wi ∈* (*T \ {t*ˆ*}*)*∗* then **m***−*

*i−*1

+ **m**+ *∈* **R**(*N'*). Then from **m***i−*1(*p*1) = 0 and

**m***i*(*p*1) = 0 it follows that **m***−*

+ **m**+ *∈* **R**(*tp* (*N'*)). Otherwise, if *wi* = *t*ˆ, we have

*—*

**m**

*i*

*i−*1

+ **m**+ in **R**(*t*ˆ).

*i−*1 *i* 1

Concatenating all with the operator *∗P\{p }* leads to **m***−* + **m**+

is in **R**(*N* ),

1 0 *e*

which means (**m***−* + **m**+)*◦*ˆ*P* +*∪P−* **R**(*N* ) = *{∅}*.

0 *e*

The other direction follows simply by composing firing sequences.

## Example:

Consider the Petri net *N*

***,,***

***) )***

*t*7

*t*8

*p*1

3 2

***J***

***^***

***,,*** *•*

***,7,***

*p*2 ***(*** 5

***J***

*t*ˆ ***(*** 7 *p*3

***J***

with the start marking *{p*2 *'→* 4*, p*3 *'→* 2*}* and the end marking *{p*2 *'→* 4*, p*3 *'→* 3*}*. We have **R**(*t*7) = *{p− '→* 1*, p*+ *'→* 3*}* + **I***dP* , **R**(*t*8) = *{p− '→* 2*, p*+ *'→* 1*}* + **I***dP* and

2 1 1 3

**R**(*t*ˆ) = *{p− '→* 7*, p*+ *'→* 5*}* + **I***dP\{p }*. This yields

3 2 1

*p*

**R**(*N*

*'*) = **R**((*P, {t*7*, t*8*},W |*

ˆ ˆ )) = *∗P*

*p−*

*p*+

*,*

1

*— p*+ =

*P×{t}∪{t}×P*

1 3 2 1

*p− , p*+ *,* *p− , p*+ *,* *p− , p*+ *, p*+ *,* *p− , p− , p*+ *,* *p− , p*+ *, p*+ *,* *p− , p*+ ,*∗* + **I***d*

2 *,*

3

2

1

1

3

1

2

3

1

2

1

1

1

3

1

2

1

1

1

3

2

2

2

1

2

3

2

2

2

3

3

*P*

1 *,*

and **R**(*t*

*p*1

*{*(*p−,x*)*,*(*p*+*,y*)*}*

2

3

*{p*2*,p*3*}*

2

3

(*N'*)) = **R**(*N'*)*◦*

*{∅}* =  *p− , p*+ ,*∗* + **I***d* .

We can cut the firing sequences in (*t*7 + *t*8 + *t*ˆ)*∗* = ((*t*7 + *t*8)*∗* + *t*ˆ)*∗* into parts in (*t*7 + *t*8)*∗* and *t*ˆ all starting and ending with no token on *p*1. This yields **R**(*N* ) :=

1

1

*∗{p ,p }*(**R**(*tp* (*N'*)) *∪* **R**(*t*ˆ))

2 3

*p− p*+

*,*

1

*p− p*+

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*,*

*p− p− p*+

*p− p− p*+

*p− p*+ *p*+

*p− p*+ *p*+

2 3 3 2 2

*,*

*,*

*,*

2 3 7 5 2

3 2 2

4 5 4

3 2 3

1 5 7

2 3 3

3 3 7

2 3 *, ...,*

*p− , p− , p*+ *,* *p− , p− , p*+ *, p*+ *,* *p− , p*+ *, p*+ *,* *p− , p− , p*+ *, p*+ ,*∗* + **I***d* .

2

4

3

2

3

8

2

5

3

1

2

1

3

7

3

6

2

4

3

3

2

4

3

2

2

4

3

3

*{p*2*,p*3*}*

1 6

# Nested Petri Nets as normal form for expressions

From now on we use the variables *t, T, N* as expressions describing transitions, sets of transitions and (sub-)nets. For an expression *e*, we will always define a *carrier set C*(*e*) *⊇ {a | ∃* **f** *∈* **R**(*e*)*,* **f** (*a*) *>* 0*}}*. The function **R** was in the previous section giving the reachability relation **R**(*e*) *⊆* N*C*(*e*) for an *e* of the form *t*, *N* or *T* . Now, we use **R** as the evaluation function for an expression where the expression operators

*∗P , ◦Q, ∪* and +, and the operator *∩* will always be defined on expressions such

that they commute with **R**.

Let the expression for an *elementary transition* have the form *t* = *Lt*, where *Lt* is an expression for the *linear set* **L***t* = **R**(*Lt*) = **c***t* + Γ*∗* described by a (constant) multiset **c***t* and a finite set of (period-) multisets Γ*t*. For example, in Sections [2.1](#_bookmark4) and [2.3](#_bookmark8), we have Γ*t* = *{{p− '→* 1*, p*+ *'→* 1*} | p ∈ P}* leading to Γ*∗* = **I***dP* . We have

*t*

*t*

*C*(*t*) := *C*(*Lt*) := *P− ∪ P* + *∪ {a | ∃* **f** *∈ {ct}∪* Γ*t,* **f** (*a*) *>* 0*}*.

Let the expression for *sets of transitions* be *T* = *t*1 *∪ t*2*... ∪ tl* for expressions for transitions *ti ∈ T* for *i ≤ l*, and the expression for a *sub-net* with places *PT* and transitions *T* be *N* = *∗PT* (*T* ). Let *C*(*N* ) := *C*(*T* ) := *t∈T C*(*t*).

Let the expression for a *generalized transition* have the form *t* = *Lt◦*ˆ*At Kt*, where

*Lt* again expresses a linear set, and *Kt* is a set of sub-nets and interpreted as

Σ

expression *Kt* = *Ni* where the *C*(*Ni*) are pairwise disjoint and *At* := *C*(*Kt*) :=

*Ni∈Kt*

*Ni∈Kt C*(*Ni*).

We define *C*(*t*) := *{a | ∃* **f** *∈ {ct} ∪* Γ*t,* **f** (*a*) *>* 0*} \ At*. This means that the

behavior of *t* is mainly described by the linear set **c***t* + Γ*∗* additionally controlled by the reachability in the sub-nets *Ni*.

*t*

*⊆* N*C*(*t*)*∪At* but it is

For example, the reachability question for a complete net in Lemma [2.1](#_bookmark5) and Lemma [2.4](#_bookmark9) is formulated as the control by a (sub-)net in *t* with **R**(*t*) = (**m***−* +

0

**m**+)*◦*ˆ*P* +*∪P−* **R**(*N* ) respectively **R**(*t*) = (**m***−* + **m**+)*◦*ˆ + *−*

+ *−* **R**(*N* ) in a way

*e* 0 *e P ∪P*

*\{p*1 *,p*1 *}*

that already has this normal form for expressions with *ct* = (**m***−* + **m**+) and Γ*t* = *∅*.

0 *e*

However, the behavior on the outside is trivial (*∅* or *{∅}*) because *C*(*t*) = *∅* in this

case. The same holds for (**m***−* + **m**+)*◦*ˆ + *− ∗P*

(*Tg−*1) in Theorem [5.1](#_bookmark27).

0 *e PTg−*1 *∪PTg−*1

*Tg−*1

The decision algorithm below will start with an expression of the form *T* = *{t}* by keeping in mind that, according to Lemma [2.4](#_bookmark9), **R**(*T* ) = **R**(*t*) = *{∅}* if there is a firing sequence *w ∈ T∗* with **m**0[*w⟩***m***e*. Otherwise **R**(*T* ) = **R**(*t*) = *∅* if there is not.

An example for the general case is **R**(*tp* (*N'*)) := *{***r** *∈* **R**(*N'*) *|* **r**(*p−*) = **r**(*p*+) =

1 1 1

0*}* form Section [2.3](#_bookmark8). Here we need Lemma [3.1](#_bookmark11) to construct the expression *t{p }*(*N'*)

1

of the desired normal form with **R**(*tp* (*N'*)) = **R**(*t{p }*(*N'*)). Again, *th* in Theorem

1 1

[5.1](#_bookmark27) has the normal form for expressions.

**Lemma 3.1** *Let N be an expression for a subnet. Then, we can construct an equivalent expression for a transition t*(*N* ) *with* **R**(*t*(*N* )) = **R**(*N* ) *and tP '* (*N* ) *with* **R**(*tP '* (*N* )) = *{***m** *∈* **R**(*N* ) *| ∀p ∈ P '* **m**(*p−*) = **m**(*p*+) = 0*}.*

**Proof.** Define *t*(*N* ) by **c***t*(*N*) := **c***t '* (*N* ) := *∅*, Γ*t*(*N*) := *{{q '→* 1*, q*ˆ *'→* 1*} | q ∈*

*P*

*C*(*N* )*}* and *Kt*(*N* ) := *{N*ˆ*}* where *N*ˆ is the result of replacing all occurrences of some

*q ∈ C*(*N* ) in *N* by *q*ˆ. This means that we make the *C*(*N*ˆ) disjoint to *C*(*t*(*N* )).

The restriction of places in *P '* to 0 is done by Γ*t* (*N*) := *{***m** *∈* Γ*t*(*N*) *| ∀p ∈*

*P'*

*P '* **m**(*p−*) = **m**(*p*+) = 0*}*.

**Example** (continued):

We identify *t*7 = *{p*ˆ*− '→* 1*, p*ˆ+ *'→* 3*}* + **I***d{p*ˆ *,p*ˆ *,p*ˆ *}*, *t*8 = *{p*ˆ*− '→* 2*, p*ˆ+

*'→* 1*}* +

2 1 1 2 3 1 3

**I***d{p*ˆ *,p*ˆ *,p*ˆ *}* and *t*ˆ = *{p−*

*'→* 7*, p*+

*'→* 5*}* + **I***d{p ,p }*. This yields the expressions

1 2 3 3 2 2 3

*T*1 = *t*7 *∪ t*8 and *N*1 = *∗{p*ˆ1*,p*ˆ2*,p*ˆ3*}*(*T*1). On the next level, we get the generalized transition *t*2 = *tp* (*N'*)

1

*∅* + *p− , p*ˆ*−* *,* *p− , p*ˆ*−* *,* *p*+ *, p*ˆ+ *,* *p*+ *, p*ˆ+ ,*∗* *◦*ˆ *N ,*

2

2

3

3

2

2

3

3

*{p*ˆ*−,p*ˆ*−,p*ˆ+*,p*ˆ+*}*

1

1

1

1

1

1

1

1

1

2

3

2

3

which we visualize as ***,***

***,,***

*t*7

*t*8

***) )***

3 *p*ˆ1 2

***J***

*t*2

= ***^***

***,,***

*p*ˆ2

***J***

***,7,***

*p*ˆ3

***J***

***J***

*T*2 = *t*2 *∪ t*ˆ and *N*2 = *∗{p ,p }*(*T*2). On the top level, we get

2

*T* = *t*

3

3

= *p− , p− , p*+ *, p*+ *◦*ˆ *N ,*

3

2

3

2

3

*{p−,p−,p*+*,p*+*}*

2

4

2

4

3

2

3

2

3

which we visualize as follows:

***,***

z ***,,***

***^ /***

2***////***

y

4***\\€,/,/***

***)*** *p*3

*t*2

***// / \J***

***\***3

***/***

7 ***\\€***

***\***

***\***

***/// /***

*p*2 ***(***

***/// 7*** 4

*t*ˆ

***J***5

***J***

* 1. *The property T*

In order to decide the emptiness problem for expressions, we want to establish a normal form *T* , which corresponds to the condition Θ in [[12](#_bookmark42)]:

**Definition 3.2** An expression *T* has the property *T* if *∀t ∈ T, ∀Ni* = *∗PT* (*Ti*) *∈ Kt*

*i*

the following conditions hold:

* + 1. In recursive manner, *Ti* has
       1. the property *T* , and
       2. For all *t' ∈ Ti* it holds *∀***g** *∈ {***c***t' }∪* Γ*t' ∃w***g** *∈ C*(*t'*) **g**(*w***g**) = 1*,*

*∀***g***' ∈ {***c***t' }∪* Γ*t' \ {***g***}* **g***'*(*w***g**) = 0.

*t'∈Ti*

This condition says that the number of times where **g** is used is exactly the number of occurrence of the *witness (place) w***g**.

Σ

* + 1. *∀***g** *∈ {***c***t}∪* Γ*t, ∀p ∈ PTi*

*'*

**g**(*p−*) *−* i*nd*(**g**)(*p−*) = **g**(*p*+) *−* i*nd*(**g**)(*p*+), where

i*nd*(**g**) :=

*t'∈Ti,***g***'∈{***c***t' }∪*Γ*t'*

**g**(*w***g***'* )**g**

describes the indirect effect of **g** using the property about the witness places in Condition 1 in the recursion for *Ti*. This property says that **g**(*w***g***'* ) is exactly the number of times that **g***'* is used. Thus, i*nd*(**g**) contains a quantitative information about the firing sequences which are allowed by **g**. The condition says that (disregarding the real control by the sub-net *Ni*) the quantitative information is consistent with the expected control.

* + 1. *∀w ∈ C*(*Ni*) *\* (*P* + *∪ P−*) Σ **g**(*w*) *>* 0. This condition says that each witness

*Ti Ti*

*g∈*Γ*t*

appears in a period and, thus, the use of each interior transition and period is unlimited.

* + 1. There are multisets *∃***m**+*,* **m***− ∈* **R**(*Ni*) with *∀p ∈ PTi*

**m**+ *|*

*— ∈* (**c***t* + Γ*∗*) *| − ∧*((*∀***g** *∈* Γ*t* **g**(*p−*) = 0) *→* **m**+(*p*+) *>* **m**+(*p−*))*∧*

*PTi t PTi*

**m***− |*

+ *∈* (**c***t* + Γ*∗*) *|* + *∧*((*∀***g** *∈* Γ*t* **g**(*p*+) = 0) *→* **m***−*(*p−*) *>* **m***−*(*p*+))*.*

*PTi t PTi*

This condition says that there is a firing sequence in the sub-net *Ni* quantita-

tively described by **m**+. This firing sequence starts with a marking available by **c***t* + Γ*∗* and increases all those places which cannot be increased by Γ*t*.

*t*

* + 1. **c***t |C*(*t*)*∈* **R**(*t*). This condition says that transition *t* can fire without the use of one of its periods in Γ*t*.
  1. *The size of an expression*

To prove the termination of the algorithm in Section [4](#_bookmark16), we have to define an ordering on a size *S* which is Noetherian and decreasing in every step of the algorithm:

A list (tuple, respectively) is smaller than another if the first *i* elements are equal and the *i* + 1’th element is smaller (or not existing). A multiset **m** is smaller than a multiset **m***'* if there is an *e* with **m**(*e*) *<* **m***'*(*e*) and **m**(*e'*) = **m***'*(*e'*) for all *e' > e*. (Thus multisets may as well be interpreted as a descending ordered list using lexicographic order.)

The smallest size is *S*(*∅*). Accordingly, if *T* = *∅* then *T* has the property *T* .

Σ

The size *S*(*T* ) = *{S*(*t*) *'→* 1*}* is a multiset of all sizes *S*(*t*) with *t ∈ T* . The

*t∈T*

size of *t* is *S*(*t*) := (*S*(*Kt*)*, b*2*, b*5 + *|*Γ*t|*). Here, *bi* = 0 if Condition *T .i* is fulfilled, and

Σ

*bi* = 1 otherwise. The size *S*(*Kt*) = *{S*(*Ni*) *'→* 1*}* of a set of nets is a multiset

*Ni∈Kt*

of the sizes *S*(*Ni*) of the nets *Ni ∈ Kt*. The size of a net is

*S*(*Ni*) := (**s***m* + *{|PTi | '→* 1*}, S*(*Ti*)*, b*1*b, |C*(*Ni*)*|*)

1*b*

with **s***m* := *max{***s** *| ∃***g***, f, b*2*, b'*

1*b*

*, e,* **s***'* **s***'*((**s***,* **g***, b'*

*,f* )) *>* 0*, S*(*Ti*)((**s***', b*2*, e*)) *>* 0*}*. In

other words, the first component is a multiset in NN which is obtained by taking the maximal of such multisets of all first components in the size of a subnet of one of the transitions in *Ti* (respectively *∅* if none exists) and adding the current number of places. The second component contains the recursion. The reason for this complicated construction comes from Section [4.4](#_bookmark22) where the recursion-depth increases but the size has to decrease. Furthermore, this causes *S*(*Ni*) to be greater than the size of its occurring subnets. This is also necessary in parts where the

Expression Carrier set

*T T .*1*.b*

**c** Γ z¬z

*t ∈ T {*

*t}∪*

*t ⊆* N

*C*(*T* ) = ˆ

⎧⎪⎨ *C*(*t*) = *P− ∪ P* + *∪{* v **g** *}*

⎪⎩ ˆ

zzz

*T*

*T*

*w***c***t ,w , ...*

*Ni ∈ Kt Ti*

*At* = *C*(*Ni*) = *C*(*Ti*) =

*T .*1*.b*  *i i i*

zzz

*w t' , w***g***, ...*

*'* **c** Γ

*t*

*∈ Ti {*

*t' }∪*

z¬z

⎧⎪⎨ *C*(*t'*) = *P− ∪ P* + *∪{* **c** v *' }*

Fig. 1. An overview over the e⎩xpr*A*es*t*s*'*i*.*o*.*n*.* s and their carrier sets.

*t' ⊆* N

*Ti*

*Ti*

algorithm works recursively since it follows that *S*(*Kt'* ) *< S*(*Kt*) for all *t'* contained one or more levels deeper in *Kt*.

**Example** (continued):

*S*(*t*7) = *S*(*t*8) = (*∅,* 0*,* 3), *S*(*T*1) = *{*(*∅,* 0*,* 3) *'→* 2*}*,

*S*(*N*1) = (*{*3 *'→* 1*}, {*(*∅,* 0*,* 3) *'→* 2*},* 1*,* 6), *S*(*t*2) = (*{S*(*N*1) *'→* 1*},* 1*,* 4),

*S*(*T*2) = *{S*(*t'* ) *'→* 1*,* (*∅,* 0*,* 2) *'→* 1*}*, *S*(*N*2) = (*{*3 *'→* 1*,* 2 *'→* 1*}, S*(*T*2)*,* 1*,* 4).

2

**Lemma 3.3** *The ordering on S deﬁned above is Noetherian*

**Proof.** As shown in [[4](#_bookmark33)], the set of descending ordered lists of elements of a Noethe- rian ordered set is again Noetherian. The first components of the quadruples *S*(*N* ) are descending lists of natural numbers and, thus, Noetherian.

Assume by contradiction that **x** is the smallest first component such that there is an infinite descending sequence of quadruples

*S*(*N* ) = (**x***,* **y**1*, b' , n*1)*,* (**x***,* **y**2*, b' , n*2)*, ....*

1 2

In all quadruples appearing in all lists in all triples appearing in any **y***i*, the first component must always be smaller than **x** and, therefore, their order must be Noetherian. Thus, the lists which are the first components of the triples are also ordered Noetherian. Since the other components are natural numbers, the triples and the **y***i*’s are also ordered Noetherian. Since the first component **x** must re- main constant, and the third and forth components are natural numbers, we get a contradiction; thus, *S*(*T* ) is Noetherian.

* 1. *Additional operators working on expressions*

The following lemma is used to restrict the semilinear part in a transition *t* as it will be needed to establish the property *T .*2

**Lemma 3.4** *Let t* = *Lt◦*ˆ*At Kt be an expressions for a transition and L be (an expression for) a semi linear set. Then, we can construct an expression T '* = *t|L (with* **R**(*T'*) = (**R**(*Lt*)*∩***R**(*L*))*◦*ˆ*A* **R**(*Kt*)*) where the occurring sizes S*(*t'*) *with t' ∈ T ' can increase relatively to S*(*t*) *only in the last position in the triple.*

*t*

**Proof.** Using Presburger arithmetic [[8](#_bookmark35)],[[5](#_bookmark36)], we can calculate for every *t ∈ T* the semi-linear set

*l*

*Lt ∩ L* =: *Lj*

*j*=1

resulting in finitely many linear sets *Lj*, and define *T '* := *t|L* := *{Lj◦*ˆ*At Kt | j ≤ l}*.

An Example is given in Subsection [4.2](#_bookmark20) where *Lt ∩ L* consists of only one linear set and thus *T '* consists of only one transition.

The following Lemma [3.5](#_bookmark14) and Lemma [3.1](#_bookmark11) allow us to bring every expression into the normal form as nested Petri nets:

**Lemma 3.5** *Let T and T ' be expressions for sets of transitions, and Q be a relation. Then, we can construct an expression T ''* := *T◦QT' (with* **R**(*T''*) = **R**(*T* )*◦Q***R**(*T'*)*)*

*where the occurring sizes S*(*t*) *can increase only in the last position in the triple and sum up in the ﬁrst position.*

**Proof.** We may assume that *At*,

*t∈T '*

(otherwise replace elements by copies). We define

*t∈T*

*At* and *π*1(*Q*) *∪π*2(*Q*) are pairwise disjoint

*T ''* := *Lj◦*ˆ*A ∪A* (*Kt* + *Kt'* )

*t t'*

*t ∈ T, t' ∈ T ',j ≤ r , Lt◦QLt'* =:

*r*

*Lj*

*j*=1

holds **R**(*T* )*◦Q***R**(*T'*) =

using Presburger arithmetics (since *Lt◦QLt'* is semilinear see Subsection [1.3](#_bookmark3)). It

*t∈T,t'∈T '*

**R**(*t*)*◦Q***R**(*t'*) =

((**L***t◦*ˆ*A* (**R**(*N*1)+ *...*))*◦Q*(**L***t' ◦*ˆ*A '* (**R**(*N'* )+ *...*))) =

*t*

*t*

1

*t∈T,t'∈T '*

((**R**(*N*1)+ *...*)*◦*ˆ *−*1 **L***t◦Q***L***t' ◦*ˆ*A '* (**R**(*N'* )+ *...*)) =

*t∈T,t'∈T '*

*At*

*t*

1

(**L***t◦Q***L***t'* )*◦*ˆ*A ∪A '* (**R**(*N*1)+ *...* + **R**(*N'* )+ *...*) = **R**(*T ''*)

*t*

*t*

1

*t∈T,t'∈T '*

since *At*, *At'* and *π*1(*Q*) *∪ π*2(*Q*) are pairwise disjoint. (see Subsection [1.2](#_bookmark2).)

Examples are given in Subsections [4.3](#_bookmark21) and [4.4](#_bookmark22).

From this we can conclude some decidability result for the first order formulas with PLUS and the *monotone transitive closure mTC* defined as follows:

Given a formula *φ*(*x*1*, ..., xk, x' , ..., x'* ), then mTC(*φ*) denotes the smallest set

1 *k*

*S ⊂* N2*k* containing all of the following:

* (*x*1*, ..., xk, x*1*, ..., xk*) for (*x*1*, ..., xk*) *∈* N*k* (this stands for the identity),
* (*x*1*, ..., xk, x' , ..., x'* ) for *φ*(*x*1*, ..., xk, x' , ..., x'* ),

1 *k* 1 *k*

* (*x*1*, ..., xk, x'', ..., x''*) for (*x*1*, ..., xk, x' , ..., x'* )*,* (*x' , ..., x' , x'', ..., x''*) *∈ S*, and

1 *k* 1 *k* 1 *k* 1 *k*

* (*x*1 + *x'', ..., xk* + *x'', x'* + *x'', ..., x'* + *x''*) for a (*x*1*, ..., xk, x' , ..., x'* ) *∈ S* and

1 *k* 1 1 *k k* 1 *k*

(*x'', ..., x''*) *∈* N*k*.

1 *k*

**Corollary 3.6** *The emptiness and satisﬁability is decidable for formulas with an FO+PLUS-formula inside and ∧, ∨, ∃ and mTC operators outside.*

**Proof.** We can express linear sets by a *t* and, thus, semilinear sets by a *T* . Now, observe that the operators work on expressions of the form *T* as follows: We can express *∧* corresponding to *∩* with *◦Q* (see Section [1.2](#_bookmark2)) and apply Lemma [3.5](#_bookmark14). For

*∨* this follows simply from *T* being already a union. The existential quantifier is done by removing the element (thus, releasing the control from the outside) and the operator mTC is done by using Lemma [3.1](#_bookmark11). Then we construct *T '* according to Theorem [4.1](#_bookmark17). According to Condition 5 of property *T* , **R**(*T* ) = **R**(*T'*) is empty if and only if *T '* = *∅*.

# The main algorithm

The idea of the algorithm is to reduce *T* if one of the conditions is not fulfilled. For Condition 2, Presburger arithmetics is used to transfer the implicit quantitative restriction by the witness places to the explicit restriction of the transitions. Condi- tion 3 ensures that all quantitative controls are unlimited. Condition 4 ensures that all places are unlimited. A covering graph construction deciding Condition 4 uses the algorithm recursively (like for Condition 1) for every step. Here, the current marking of a node is being included as a restriction to the semilinear set. Limited places are deleted at the cost of a larger structure. This larger structure, however, contains parts which are generated by restricting parts. This restriction might cause them to loose the property *7* reached by a previous recursive step. However, as we will see because of their smaller size, the property *7* can be established again and the whole algorithm will still terminate.

**Theorem 4.1** *For every expression T , we can effectively construct a T ' with*

**R**(*T* ) = **R**(*T'*) *such that T ' has property 7 .*

**Proof.** The expression *T '* in the Theorem is computed by the following algorithm where the details are explained in the subsections:

**function** reacheq(*T* ):

## begin

**repeat**

i:= 1

**while** i*≤*5 and *6t ∈ T, 6N ∈ Kt* Condition *7* .i fulfilled

**do** i:=i+1 **od**

**if** i=6 **then return** *T*

**else** *T* :=*T'* for *T '* according to subsection [4](#_bookmark16).i **fi until** i=6

**end** reacheq

in each step *S*(*T* ) decreases (*S*(*reacheq*(*T* )) *< S*(*T* ) if *T /*= *reacheq*(*T* )); due to Lemma [3.3](#_bookmark12) the algorithm terminates.

The following table shows how the size *S*(*t*) can change during the steps of Chapter [4](#_bookmark16):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *S*(*t*) | | | | | | | |
| *S*(*Ni*) | | | | | | *b*2 | *b*5 + *|*Γ*t|* |
| **s***m* + *{|PTi | '→* 1*}* | *S*(*t'*) for *t' ∈ Ti* | | | *b*1*b* | *|C*(*Ni*)*|* |
| *S*(*Kt'* ) | *b*2 | *b*5 + *|*Γ*t' |* |
| [4.1](#_bookmark19) | - | - | - | - | *↓ ↑* | | *↑* - | |
| [4.2](#_bookmark20) | - | - | - | - | - - | | *↓ ↑* | |
| [4.3](#_bookmark21) | - | - | - | - | - *↓* | | *↑ ↑* | |
| [4.4](#_bookmark22) | *↓* | *↑* | *↑* | *↑* | *↑ ↑* | | *↑ ↑* | |
| [4.5](#_bookmark23) | - | - | - | - | - - | | - *↓* | |

**Corollary 4.2** *The reachability problem for a Petri net with one inhibitor arc is decidable.*

**Proof.** According to Lemmata [2.4](#_bookmark9) and [3.1](#_bookmark11), we can construct an expression *T* where **R**(*T* ) = *{∅}* (and is not empty) if and only if there is a firing sequence *w ∈ T∗* with **m**0[*w⟩***m***e*. Then, we construct *T '* according to Theorem [4.1](#_bookmark17). According to Condition 5 of property *7* , **R**(*T* ) = **R**(*T'*) is empty if and only if *T '* = *∅*.

**Remark:** Alternatively, instead of using Lemma [2.4](#_bookmark9), we can construct *T* for a given net *N* = (*P, T, W, {*(*p*1*, t*ˆ)*},* **m**0*,* **m***e*) directly by connecting three sub-nets with a linear set in such a way that the first sub-net expresses firing sequences in (*T \ {t*ˆ*}*)*∗* starting with **m**0 before the first firing of *t*ˆ, the second contains only one transition which has a sub-net itself expressing firing sequences *∈* (*t*ˆ(*T \ {t*ˆ*}*)*∗*)*∗* ending with markings without tokens on *p*1 and the third sub-net expresses firing sequences in (*T \ {t*ˆ*}*)*∗* ending with **m***e*.

* 1. *Condition 1 Recursion and introducing witnesses*

Let Condition 1 be not fulfilled by *Ti*; let *T '* := *reacheq*(*Ti*), which terminates by

*i*

induction since *S*(*Ti*) *< S*(*T* ).

For all *tj ∈ T '* let *Gj* be the set of all **g** *∈ {***c***t }∪* Γ*t*

not having a witness. Add

*i j j*

witnesses*{w***g***' |* **g** *∈ Gj}* to *C*(*t'* ) := *C*(*tj*) *∪ {w***g***' |* **g** *∈ Gj}* by replacing each **g** in

*j*

*Gj* by **g***'* := **g** + *{w***g***' '→* 1*}* in

⎧

⎨ *tj ∈ T ', At'*

= *At , Kt'*

= *Kt ,* ⎫⎬

*T ''* := *t'*

*i*

*j*

⎩

*t*

Γ *'*

*j*

= Γ*tj \ Gj ∪ {***g**

Now, we set *t'* := *Lt' ◦*ˆ*A ' Kt'* with *Kt'* = *Kt \ {Ni}∪ {N ''}*, *N''* = *∗P*

*i j j j j .*

*'* ⎭

*|* **g** *∈ Gj \ {***c***t}* and **c***t'*

:= **c***tj* if **c***tj ∈/ Gj*

*j*

*t i i T* *i*

*''* (*T ''*),

*i*

*At'* = *At ∪ C*(*N''*) and Γ*t'* := Γ*t ∪ {{w '→* 1*} | w ∈ C*(*N''*) *\ C*(*Ni*)*}*, and let

*i* *i*

*T '* := *T \ {t}∪ {t'}*. Since **R**(*Ti*) = **R**(*T ''*) *|C*(*T* ), we have **R**(*Ni*) = **R**(*N''*) *|C*(*T* );

*i i i* *i*

thus, **R**(*t*) = **R**(*t'*); thus, **R**(*T* ) = **R**(*T'*).

Since *S*(*tj*) = *S*(*t'* ) for all *tj ∈ T '*, the size *S*(*T ''*) = *S*(*T'*) remains unchanged.

*j i i* *i*

The only increase was *|C*(*N''*)*| ≥ |C*(*Ni*)*|* but but we have either *S*(*T ''*) = *S*(*T'*) *<*

*i i* *i*

*S*(*Ti*), or in case *T '* = *Ti*, we have now *b*1*b* = 0. From that follows that *S*(*N''*) *<*

*i* *i*

*S*(*Ni*); thus, *S*(*t'*) *< S*(*t*) and *S*(*T'*) *< S*(*T* ).

**Example** (continued):

Since the expression *T*1 does not fulfill Condition 1, we add the two witnesses *w***c** *'*

*t*7

and *w***c** *'* . For simplicity, we omit the witnesses for the periods for **I***dP* in the

*t*8

elementary transitions. So we replace *t*7 and *t*8 by *t'* = *{p*ˆ*− '→* 1*, p*ˆ+ *'→* 3*, w***c** *'→*

7 2 1 *'*

*t*

7

1*}* + **I***dp*ˆ *,p*ˆ *,p*ˆ

and *t'*

= *{p*ˆ*−*

*'→* 2*, p*ˆ+

*'→* 1*, w***c**

*'→* 1*}* + **I***dp*ˆ *,p*ˆ *,p*ˆ . This yields

1 2 3 8 1 3 *'*

*t*

8

1 2 3

the expressions *T ''* = *t' ∪ t'* and *N''* = *∗{p*ˆ *,p*ˆ *,p*ˆ *}*(*T ''*). On the next level, we get

1 7 8 1 1 2 3 1

*t'* = (*∅* + *{{p− '→* 1*, p*ˆ*− '→* 1*}, {p− '→* 1*, p*ˆ*− '→* 1*}, {p*+ *'→* 1*, p*ˆ+ *'→* 1*}, {p*+ *'→* 1*, p*ˆ+ *'→*

2 2 2 3 3 2 2 3 3

1*}, {wc ' '→* 1*}, {wc ' '→* 1*}}∗*)*, ◦*ˆ *− −* + + *N''* for the generalized transition and

*t*7 *t*8

*T '* = *t' ∪ t*ˆ.

*{p*ˆ2 *,p*ˆ3 *,p*ˆ2 *,p*ˆ3 *}* 1

2 2

The new sizes are now *S*(*t'* ) = *S*(*t'* ) = (*∅,* 0*,* 3) = *S*(*t*7),

7 8

*S*(*T ''*) = *{*(*∅,* 0*,* 3) *'→* 2*}* = *S*(*T*1),

1

*S*(*N''*) = (*{*3 *'→* 1*}, {*(*∅,* 0*,* 3) *'→* 2*},* 0*,* 8) *< S*(*N*1),

1

*S*(*t'* ) = (*{S*(*N''*) *'→* 1*},* 1*,* 6) *< S*(*t*2),

2 1

*S*(*T'*) = *{S*(*t'* ) *'→* 1*,* (*∅,* 0*,* 2) *'→* 1*} < S*(*T*2).

2 2

* 1. *Condition 2 Quantitative consistency*

Let Condition 2 be not fulfilled by *Ti*. The set **L** :=

**g** *∈* N*C***L** *6p ∈*

*Ni∈Kt*

*PTi*

**g**(*p−*) *−* i*nd*(**g**)(*p−*) = **g**(*p*+) *−* i*nd*(**g**)(*p*+) ,

on the carrier set *C***L** = *C*(*t*) *∪ C*(*Ni*) is a Presburger set. Since **R**(*t*) *⊆*

*Ni∈Kt*

**L***|C*(*t*) follows from the definition of **R**(*t*) and the function i*nd*, we can set *T '* :=

*T \ {t}∪ t|L* using Lemma [3.4](#_bookmark13). In other words, we have cut something away which could not have been in **R**(*T* ) anyway.

Since *b*2 is now 0 for each *tj ∈ t|L* and *S*(*Ktj* ) remains the same as *S*(*Kt*), according to Lemma [3.4](#_bookmark13), it holds *S*(*T'*) *< S*(*T* ).

**Example** (continued):

We see that *t'* does not fulfill Condition 2 when we look at the resulting equation

2

*− − −* + + +

**g**(*p*ˆ ) *−* **g**(*w***c** *'* )**c***t'* (*p*ˆ ) *−* **g**(*w***c** *'* )**c***t'* (*p*ˆ ) = **g**(*p*ˆ ) *−* **g**(*w***c** *'* )**c***t'* (*p*ˆ ) *−* **g**(*w***c** *'* )**c***t'* (*p*ˆ )

*t*7 7

*t*8 8

*t*7 7

*t*8 8

for all *p ∈ P* characterizing *L*. This is equivalent to the following three equations:

*t*

2**g**(*w***c**

) = 3**g**(*w***c** ), **g**(*p*ˆ*−*) *−* **g**(*w***c**

) = **g**(*p*ˆ+), **g**(*p*ˆ*−*) = **g**(*p*ˆ+) *−* **g**(*w***c**

). Their

*' '* 2

*t*

*t*

8 7

*'* 2 3 3 *'*

7 8

*t*

solutions are described by the linear set *Lt''* = *Lt' ∩ L* =

*t'*

*p−*

*∅* +

2 *,*

1

*p*ˆ*−*

*p*+ *p*ˆ+

*p−*

*p*ˆ*−*

2

*p*+ *p*ˆ+

3 *,*

1

3 *,*

1

3

1

2

*w***c** *w***c**

*t'*

*,*

2 7 *,*

3 8 *,*

*p− p*ˆ*−*

*p− p*ˆ*−* *∗*

and yield *t''*

2 *,*

1

2 *,*

1

2

1

*,*

3 *,*

1

2 *,*

2

2 *,*

2

3 *,*

3

3

3

= *t' |L* = *Lt'' ◦*ˆ

*−* + *−*

+ *N''*

with *S*(*t''*) = (*{S*(*N''*) *'→* 1*},* 0*,* 3) *<*

2 2 2 *{p*ˆ2 *,p*ˆ2 *,p*ˆ3 *,p*ˆ3 *}* 1 2 1

*S*(*t'* ). Since *T ''* = *t'' ∪ t*ˆ fulfills the remaining properties, we can continue one level

2 2 2

higher.

Adding the witnesses leads to *Lt'''* =

2

+ *p− , p*ˆ*− , p*+ *, p*ˆ+ *, w*1 *,* *p− , p*ˆ*− , p*+ *, p*ˆ+ *, w*2 *,* *w***c** *' , w***c** *' , p− , p*ˆ*− , p− , p*ˆ*− , w*3 ,*∗*

*∅*

2

1

2

1

2

1

2

1

1

3

1

3

1

3

1

3

1

1

*t*7

2

*t*8

3

2

2

2

2

3

3

3

3

1

(we omit the witness for *∅*.) with *S*(*t'''*) = *S*(*t''*) = (*{S*(*N''*) *'→* 1*},* 0*,* 3).

2 2 1

Defining *T '''*

= *t''' ∪ t*ˆ*'* with *S*(*T '''*) = *S*(*T ''*) and *N'''*

= *∗{p*ˆ *,p*ˆ *,p*ˆ *}*(*T '''*) with

2 2 2 2 2 1 2 3 2

*S*(*N'''*) = (*{*3 *'→* 1*,* 2 *'→* 1*}, S*(*T '''*)*,* 0*,* 8) *< S*(*N''*) = (*{*3 *'→* 1*,* 2 *'→* 1*}, S*(*T '''*)*,* 1*,* 4)

2 2 2 2

we get

*t'* = *p− , p− , p*+ *, p*+ + *w*1 ! *,* *w*2 ! *,* *w*3 ! *,* *w***c** *'* ,*∗* *◦*

3

2

3

2

3

1

1

1

*t*ˆ

*{p−,p*+*,p−,p*+*}*

2

4

2

4

3

1

2

*N'''.*

2

3

3

Establishing Condition 2 leads to

2

*t''* = *p− , p− , p*+ *, p*+ *, w*3 *, w***c** *'* + *w*1 ! *,* *w*2 !}*∗* *◦*

3

2

3

2

3

5

*t*ˆ

1

1

*{p−,p*+*,p−,p*+*}*

4

2

4

3

2

2

*N'''.*

2

3

3

* 1. *Condition 3 Elimination of witnesses*

Let Condition 3 be not fulfilled by witness *w ∈ C*(*Ni*) *\* (*P* + *∪ P−*). This means

*Ti Ti*

that we can replace *Ni* by some expression *T*ˆ with **R**(*T*ˆ) = **R**(*Ni*)*◦*ˆ*{w}***c***t|w* since for

all **m** *∈* **L***t*, we have **m**(*w*) = **c***t*(*w*). Then, we can set

*T '* := *T \ {t}∪* (*Lt | ◦*ˆ*A \C*(*N* )(*Kt \ {Ni}*))*◦*ˆ*C*(*N* )*\{w}T*ˆ

*{w} t i* *i*

which repaces the transition *t* = *Lt◦At Kt* by all those sets of transitions which result from using Lemma [3.5](#_bookmark14) (because *T*ˆ is not a net). This means that *Ni* is removed

and the equivalent *T*ˆ is plugged in at the same range; thus, **R**(*t*) = **R**(*Lt*) *|{w}*

*◦*ˆ*A \{w}*(**R**(*Kt \ {Ni}*)+ **R**(*T*ˆ)).

*t*

To create *T*ˆ = *Tγ*, we consider every possible combination *γ* (including the

*γ*

*lγ*

Σ

order of the summands) of **c***t*(*w*) =

*m*=1

**g***m*(*w*) with **g***m ∈* **c***tm* + *{***g** *∈* Γ*tm |* **g**(*w*) *>*

0*}∗*, **g***m*(*w*) *>* 0 and *tm ∈ Ti* and build *t'*

*m*

with *Lt'* = **g***m |{w}* +*{***g** *∈* Γ*tm |* **g**(*w*) =

0*}∗* and *Kt'* := *Kt* . The expressions *t'*

*m*

*m*

*m*

*m*

describe the parts in which *w* was used.

In *N'* = *∗P '* (*T '*) with *T '* :=

*i T i* *i*

*i*

*{t | t ∈ Ti,* **c***t''* = **c***t''' , Kt'''* = *Kt'' ,* **c***t''* (*w*) = 0*,* Γ*t'''* = *{***g** *∈* Γ*t'' |* **g**(*w*) = 0*}},*

*''' ''*

we filter out everything which affects *w*; thus, *C*(*N'*) = *C*(*T'*) = *C*(*Ti*) *\ {w}* and

*i* *i*

**R**(*N'*) = *{***m** *∈* **R**(*Ni*) *|* **m**(*w*) = 0*}*. Then, using Lemma [3.1](#_bookmark11), we construct *t*(*N'*)

*i* *i*

which has now the property **R**(*t*(*N'*)) = *{***m** *∈* **R**(*Ni*)*|***m**(*w*) = 0*}*. Now, we define

*i*

*i*

*Tγ* := *t*(*N'*)*◦P*

*t' ◦P*

*t*(*N'*)*◦P t' ◦P*

*...◦P*

*t' ◦P t*(*N'*)

*i*

again using Lemma [3.5](#_bookmark14).

*T '* 1 *T'*

*i* *i*

*i T'* 2 *t i*

*T ' lγ T '* *i*

*i* *i*

It holds *S*(*Kt'* ) *< S*(*Kt*) for every new *t'* in *T '* because of *S*(*N'*) *< S*(*Ni*). This

*i*

in turn follows from *|C*(*T '*)*|* = *|C*(*Ti*)*|−* 1 and *S*(*N* ) *< S*(*Ni*) for all *N ∈ Kt* , and

*i m*

*m ≤ lγ* for all *γ*.

It holds *S*(*T'*) *< S*(*T* ) since *S*(*t'*) *< S*(*t*) for every *t'*.

*t*

*{p}*

*j*

**Example:** Consider *t* with **c** = *w , p− , p*+ , *6***g** *∈* Γ

*t*

2

4

5

*t*

**g**(*w*) = 0, *K*

= *{∗*

(*v∪t* )*}*,

and **c** = *w , p− , p*+ *, q− , q*+ , *K* = *{∗* (*u*)*}*.

*tj*

1

6

7

8

9

*tj*

*{q}*

***, ,,,***

*q*

***////***

*v*

***/ cc zJza***

*u*

8 9

***//***

***<***6

***/J***7

2 ***) ,,/L/ )***

*p*

4  ***J*** 5

***J***

Then *t'* is defined such that **c** = *p− , p*+ *, q− , q*+ *, p− , p*+ *, q− , q*+ *, p− , p*+ , fur-

thermore,

*,*

1

*−* + *−*

1 0 2

1 1 1

*'* 0 0 1 1

4 6 8 9

*p ,*

*p*

*p ,*

*p*

+

1 *∈* Γ*t'* and

1 1 2 2 2 2

7 6 8 9 7 5

*t*

*Kt* = *{∗{p*0*}*(*v*0)*, ∗{q*1*}*(*u*1)*, ∗{p*1*}*(*v*1)*, ∗{q*2*}*(*u*2)*, ∗{p*2*}*(*v*2)*}*, where *pi, qi, vi* and *ui*

are replacements caused by disjointness condition in Lemma [3.5](#_bookmark14).

***, , , , ,***

***,, ,, ,, ,, ,,***

*p*0 ***a***

*q*1 ***a***

*p*1 ***a***

*q*2 ***a***

*p*2 ***a***

***J J***

***J***

6+y

***J*** 7+y ***J***

4 6+x 8 9 7+x 8 9 5

*v*0

*u*1

*v*1

*u*2

*v*2

***J J J J J***

The variables *x* and *y* illustrate the effect of the periods in Γ*t'* which originate from the (omitted) periods of *tj*.

* 1. *Condition 4 Elimination of bounded places*

Condition 4 is decidable by two *covering graph* constructions for every *i* working as follows: Every node in the covering graph *CG*(*i,*+) (*CG*(*i,−*), respectively ) has a

*P*

marking from (N *∪ {ω}*)

*—*

*Ti* ((N *∪ {ω}*)

+

*Ti* , respectively ). The root of the covering

*P*

graph *CG*(*i,*+)

has the marking **c***t |P−*

*Ti*

+*ω{p−|∃***g***∈*Γ**g**(*p−*)*>*0*}*.

For a node in *CG*(*i,*+) marked with **m**, we construct *T '*

*i*

*i*

with **R**(*T'*) = *{***g** *∈*

**R**(*Ti*) *|* **g** *|*

*— ≤* **m***}* using Lemma [3.4](#_bookmark13) as *T '* := *{t'|{***g***∈***L**

*|* **g***|*

*≤***m***} | t' ∈ Ti}*. This

*PTi*

*i t' P−*

*Ti*

restricts the allowed multisets to those which are possible starting with the limited marking **m**. All *Kt'* with *t' ∈ T '* appear in the subnet *Ni* in *t* (unchanged by Lemma [3.4](#_bookmark13)). For all *N' ∈ Kt'* , we have *S*(*N'*) *< S*(*Ni*) since the first component in *S*(*Ni*) is

*i*

*{|PTi | '→* 1*}* plus the maximum of everything one level deeper. Therefore, we have

*S*(*Kt'* ) *< S*(*Kt*) for all *t' ∈ T '* and, thus, *S*(*T'*) *< S*(*T* ). This allows us to compute

*i* *i*

*T ''* := *reacheq*(*T'*) recursively.

*i* *i*

For every *t'' ∈ T ''*, (since we know from Condition *7 .*5 that **c***t''* alone can fire),

*i*

we add a new node

*'*

*−* + +

**m** := **m** *−* **c***t'' |P−* +*{p*

*Ti*

*'→* (**c***t''* (*p*

)+ *ω* Σ

**g***∈*Γ*t''*

**g**(*p*

)) *| p ∈ PTi }*

to the covering graph *CG*(*i,*+). According to Corollary [4.4](#_bookmark25), there is no limit for the number of appearances of the multi-sets in Γ*t''* in firing sequences. This allows us to label those places *p−* with *ω* where **g**(*p*+) *>* 0 for a **g** *∈* Γ*t''* .

If **m***' >* **m***''* for an **m***''* on the path from the root to **m**, then we set **m***'* := **m***'* + *ω*(**m***' −* **m***''*). This is because we can lift the marking of those places *p−* with (**m***' −* **m***''*)(*p−*) *>* 0 by repeating the firing sequence corresponding to the path from **m***''* to **m***'* arbitrarily many times.

If **m***' ≤* **m***'' ∈ Path*(**m***'*), then we need not calculate the successors of **m***'* since we already had better chances at **m***''*.

According to [[3](#_bookmark34)], there are only finite sets of incomparable multi-sets over a finite set *P−*. It, therefore, follows that every path must terminate.

*T*

*i*

If for all *i* a node marked with *ω*

*P* +

*—*

*Ti* is in *CG*(*i,*+) and, analogously, a node

*P*

marked with *ω*

*Ti* is in *CG*(*i,−*), then the Condition 4 is fulfilled. Otherwise, we can

calculate without loss of generality

*k* := *min max min max* **m**(*pσ*)

*σ∈{*+*,−} path⊆CG*(*i,σ*) *p∈PTi* **m***∈path*

This means that in every path in *CG*(*i,*+) or *CG*(*i,−*), there is a place *p* such that on this path there are never more than *k* tokens on *p−* or *p*+ respectively.

Now, we can replace in *T '* := *T \ {t}∪ U* (*p*) the transition *t* by all those

*p∈PTi*

sets of transitions *U* (*p*), described in the following sub section, which are generated by restricting *t* in such a way that, in the subnet *Ni*, there can never be more than *k* tokens on *p*.

In order to show that *S*(*T'*) *< S*(*T* ) we have to show that each *S*(*t'*) *< S*(*t*) for every *t'* in every *U* (*p*).

*4.4.1 Elimination of places*

As in the construction of a regular expression from a finite automaton having the states 0*, ...k*, we define for all *l, j, h ≤ k* an expression *T l−*1 describing corresponding firing sequences with the following property: They start with a marking **m**0 with **m**0(*p*) = *j*, end with a marking **m**1 with **m**1(*p*) = *h*, and meanwhile the number tokens on *p* is always less than *l*. This allows us to remove the place *p* since its

*j,h*

information is no longer necessary. Therefore, we have *P*

*l−*1 = *P'* := *PT*

*\ {p}*).

*Tj,h Ti* *i*

For an inductive definition, we start with the case of an immediate success where there is no ’meanwhile’: This means

*T−*1 := *Ti◦*ˆ*{p−,p*+*}{{p− '→ j, p*+ *'→ h}}*

*j,h*

is constructed using Lemma [3.5](#_bookmark14). (We can write *{{p− '→ j, p*+ *'→ h}}* as *{tj,h}* with

*j,h*

**c***tj,h*

= *{p− '→ j, p*+ *'→ h}* and Γ*t*

= *Ktj,h*

= *∅*.) Recursively, we define

*T l* := *{t*(*Nl−*1)*}* := *{t*(*∗P'*

(*T l−*1))*}*

*l,l*

*l,l*

*Ti l,l*

using Lemma [3.1](#_bookmark11). Then with Lemma [3.5](#_bookmark14), we construct

*T l* := *T l ◦P'*

*T l−*1 for *h /*= *l,*

*l,h*

*l,l*

*Ti l,h*

*T l* := *T l−*1*◦P' T l*

for *j /*= *l,* and

*j,l*

*j,l*

*Ti l,l*

*T l* := *T l−*1*◦P'*

*T l ◦P'*

*T l−*1 *∪ T l−*1 for *h /*= *l ∧ j /*= *l.*

Now we define

*j,h*

*j,l*

*Ti l,l*

*Ti l,h*

*j,h*

*U* (*p*) = (*Lt |*

*−* + *◦*ˆ*A \C*(*N* )(*Kt \ {Ni}*))*◦*ˆ*C*(*N* )*\{p−,p*+*}T k −* +

*{p ,p } t* *i*

*i* **c***t*(*p*

)*,***c***t*(*p* )

using Lemma [3.5](#_bookmark14). We have *S*(*N'*) *< S*(*Ni*) for every *N' ∈ Kt'* with *t' ∈ Tk*

**c***t*(*p*

*−*)*,***c***t*(*p*+)

because for the corresponding first components *s'* and *si* of the 4-tuples, we have

**s***'*(*|P ' |*) = **s***i*(*|P ' |*)+ *k* + 1 but **s***'*(*|PT |*) = **s***i*(*|PT |*) *−* 1 (It holds *|P ' |* = *|PT |−* 1).

*Ti Ti i i Ti* *i*

Thus, *S*(*t''*) *< S*(*t*) for every *t'' ∈ U* (*p*).

**Example:** Let *t* = (**c** + Γ*∗*)*◦*ˆ*C*(*N* )*∗{p}∪P* (*Ni*) with **c**(*p−*) = 1, *Ni* = *v ∪ w ∪ tj*,

*i*

*C*(*Ni*) = *{p−, p*+*} ∪ P− ∪ P* + and *tj* = (**c***j* + Γ*∗*)*◦*ˆ*{q−,q*+*}∪Q−∪Q*+ *∗{q}∪Q*(*u*) with

*j*

**c***j*(*p*+) = 1, **c***j*(*q−*) = 8 and **c***j*(*q*+) = 9 look like

***,,,,***

*q*

***c c zJza***

*u*

8 9

***/***

***/J***

***,,/L***

***)*** *p*

***J***

*v*

*w*

***J***

and *k* = 1. Furthermore, we assume no other occurrence of *p* in any other constant or period. This means that the firing sequences are restricted to the regular expression ((*wv∗tj*) + *v*)*∗wv∗*. This corresponds to *T−*1 and *T−*1 to consist only of a copy of

0*,*0 1*,*1

*v*, *T−*1 only of a copy of *w* and *T−*1 only of a copy of *tj*.

1*,*0 0*,*1

We get *T* 0 = *t*(*∗P* (*T−*1)), *T* 0 = *T−*1*◦P T* 0 *◦P T−*1 *∪ T−*1; in the end every

0*,*0

0*,*0

1*,*1

1*,*0

0*,*0

0*,*1

1*,*1

new transition *t'* in (**c** + Γ*∗*) *| −* +

*◦*ˆ*C*(*N* )*\{p−,p*+*}T* 1

with *T* 1

= *T* 1

* *P T* 0 =

*{p ,p }* *i*

1*,*0

1*,*0

1*,*1

1*,*0

*t*(*∗P* (*T* 0

))*◦P T−*1*◦P T* 0

now looks like

1*,*1

1*,*0

0*,*0

***,***

***,***

***,,,,***

***c*** *q* ***za ,,***

*T−*1

1*,*0

*T−*1

0*,*0

8 ***J***9

*u*

*−*1

0*,*0

1*,*0

*T−*1

*T*

***J***

***J J***

*T−*1

1*,*1

***J***

* 1. *Condition 5 Making the constant ﬁring*

Σ

If Condition 5 is not fulfilled for *t* then, according to Corollary [4.4](#_bookmark25), for **f** = **g**,

**g***∈*Γ

there exists a (smallest) *k* such that (*c* + *k***f** ) *|C*(*t*)*∈* **R**(*t*). So we decompose *Lt* such

that **R**(*Lt*) = **R**(*Lt* + *k***f** ) *∪* **R**(**c***t* + *j***g** + (Γ*t \ {***g***}*)*∗*). Set

**g***∈*Γ *j≤k*

*T '* := *T \ {t} ∪ {t' | Kt'* = *Kt,* Γ*'* = Γ*t ∧* **c***t'* = **c***t* + *k***f** )*}*

*t*

*∪ {t' | ∃j ≤ k,* **g** *∈* Γ Γ*'* = Γ*t \ {***g***}*) *∧* **c***t'* = **c***t* + *j***g**)*}.*

*t*

Since Conditions 1 and 2 are not affected, *b*2 and *S*(*Kt*) do not change. The size

*S*(*t'*) is smaller than *S*(*t*) since *b*5 is now zero respectively *|*Γ *\ {***g***}| < |*Γ*|*; thus, it holds *S*(*T'*) *< S*(*T* ).

**Lemma 4.3** *If the conditions 1 - 4 hold for t, then it holds*

*6***f** *∈* Σ **g** + Γ*∗6***e** *∈* (Γ*t ∪ −*Γ*t*)*∗∃k ≥* 2 (**c***t* + *k***f** ) *|C*(*t*)*,* (**c***t* + *k***f** + **e**) *|C*(*t*)} *⊆* **R**(*t*)

*t*

**g***∈*Γ*t*

The proof is in [[17](#_bookmark47)] where we show that we can build up firing sequences which compensate the ’odd’ firing sequences resulting from condition 4, from the constant, and from the ’odd’ indirect firing sequences in order to find a **c***t* fulfilling condition 5 according to the following corollary:

**Corollary 4.4** *If the conditions 1 - 4 hold for t, then it holds*

*6***f** *∈* Σ **g** + Γ*∗∃k ≥* 2 (**c***t* + *k***f** ) *|C*(*t*)*∈* **R**(*t*)

*t*

**g***∈*Γ*t*

# The reachability relation for Petri nets with inhibitor arcs

Now, we generalize Lemma [2.4](#_bookmark9) by using the operators *∪, ◦Q* and *∗Q* over finite sets of multisets in a nested way. This allows us to express the reachability problem in a Petri net for which there exists an ordering of the places such that a place has an inhibitor arc to all those transitions which have an inhibitor arc from a preceding place:

**Theorem 5.1** *In a Petri-net* (*P, T, W, I,* **m**0*,* **m***e*) *with*

*∃***g** *∈* N*P 6p, p' ∈ P* **g**(*p*) *≤* **g**(*p'*) *→* (*6t ∈ T* (*p', t*) *∈ I →* (*p, t*) *∈ I*)*,*

+

*we can construct an expression Tg such that there is a ﬁring sequence w ∈ T∗ with*

**m**0[*w⟩***m***e if and only if* **R**(*Tg*) *is (*= *{∅} and) not empty.*

**Example:** For the Petri net

2

***,,***

***( )*** *p*1

*t*6

*t*7

***\‘\*** 3

***///***

***// J***

***,,/ ///***

***cc***

2  ***/ / μ,,***

*p*4 ***y ccc***

***// //*** *p*2

***cJc*** 5

***///// J***

***,7c,***

***)*** *t*9

*•****//***

5 ***\*** *•*

*p*3 ***(*** *• t*

2

8

***J***

with the start marking *{p*3 *'→* 3*, p*4 *'→* 2*}* and the end marking *{p*4 *'→* 27*}* we can easily find the function **g** with **g**(*p*1) = 1, **g**(*p*2) = 2 and **g**(*p*3) = **g**(*p*4) = 3. The

construction gives us an expression which we visualise as

***,***

27***//,***

***,/, ,,,***

*''*

*p*

4

2 ***/’ J***

*'*

2

*p*

***J***

***,,***

***)*** *p*1

*t*6

2

***/ ^***5

***7***5  ***J***

***/*** *t*8

*t*9

***7***2

***,\‘\*** 3

*p*4

***, 7***

***(*** *t*7

***^ ,, cJcc*** 2

*'*

*p*

3 ***,,*** 3

***,7c,***

***,7 ,***

***,J,*** *p*3 *p*2

*'' '*

*p*

*p*

3 4

***J J***

***J J***  ***J***

***J***

***J***

with the innermost level

= *p*

*p*

*p*

*p* + **I**

*— p*+ +

*T*1 4 *,* 1 *,* 3 *,*

This enables the firing sequence *w* = *t*6*t*7*t*7 from *p*3 *, p*4 ! to *p*2 *, p*4 ! on the inner-

3

2

1

*— p−* + +

2 *,* 4

1 3 *p*

*, ,*

1 1 2 1

*dP .*

*− −* + +

most level as

*p*3 *, p*4 *, p*2 *, p*4

1 3 4 2

1

3

4

2

*T*1

*p− , p*+ *∈* **R**(*T* ) for *t* , we get the firing sequence *w'* = (*w*)(*w*)*t* (*w*)*t* (*w*)*t* (*w*)*t*

*∈* **R**(*∗P*

(*T*1)) = **R**(*t*2) *⊆* **R**(*T*2). Together with

2

5

3

2

2

8

8

8

8

8

from *p*3 *, p*4 ! to *p*3 *, p*4 ! on the next level as *p− , p− , p*+ *, p*+ *∈* **R**(*∗* (*T* )) =

2

7

5

2

3

2

4

7

3

5

4

2

*PT*2

2

**R**(*t* ) *⊆* **R**(*T* ). Together with *p− , p*+ *∈* **R**(*T* ) for *t* , this enables the firing se-

3

3

3

1

4

5

3

9

quence *w''* = *t* (*w'*)*t*5 from *p*3 *, p*4 ! to *p*4 ! on the following level as *p− , p− , p*+ *∈*

**R**(*∗PT*

(*T*3)) = **R**(*t*4) = **R**(*T*4).

3

9

9

3

2

27

3

3

4

2

4

27

The general construction is in [[17](#_bookmark47)].

With Theorem [4.1](#_bookmark17) we derive the following:

**Corollary 5.2** *The reachability problem for a Petri net* (*P, T, W, I,* **m**0*,* **m***e*) *with*

*∃***g** *∈* N*P 6p, p' ∈ P* **g**(*p*) *≤* **g**(*p'*) *→* (*6t ∈ T* (*p', t*) *∈ I →* (*p, t*) *∈ I*)*,*

+

*is decidable.*

This is optimal since the non-existence of such an ordering would mean that there are two inhibitor arcs (*p, t*)*,* (*p', t'*) without (*p, t'*) or (*p', t*) being an inhibitor arc, which would already mean undecidability of the reachability problem for such instances.

# Priority-Multicounter-Automata

We define a priority-multicounter-automaton by a restrictive zero-test according to an order of the counters in the following way: the first counter can be tested for zero at any time; the second counter can only be tested for zero simultaneously with the

first counter; any further counter can only be tested for zero simultaneously with all preceding counters. Formally, this reads as follows:

A *priority-multicounter-automaton* is a one-way automaton described by the

6-tuple

*A* = (*k, Z,* Σ*, δ, z*0*, E*)

with the set of states *Z*, the input alphabet Σ, the transition relation

*δ ⊆* (*Z ×* (Σ *∪ {λ}*) *× {*0 *... k}*) *×* (*Z × {−*1*,* 0*,* 1*}k*)*,*

initial state *z*0, the accepting states *E ⊆ Z*, the set of configurations *CA* = *Z ×* Σ*∗ ×*

N*k*, the initial configuration *σA*(*x*) = *⟨z*0*, x,* 0*, ...,* 0*⟩* and configuration transition

relation

`˛*k*¸x

*⟨z, ax, n*1*, ..., nk⟩ |A ⟨z', x, n*1 + *i*1*, ..., nk* + *ik⟩*

if and only if *z, z' ∈ Z, a ∈* Σ *∪ {λ}, ⟨*(*z, a, j*)*,* (*z', i*1*, ...ik*)*⟩∈ δ*, *6i ≤ j ni* = 0.

The language recognized by an priority-multicounter-automaton *A* is *L*(*A*) =

*{w | ∃ze ∈ E ∃n*1*, ..., nk ∈* N *⟨z*0*, w,* 0*, ...,* 0*⟩ | ∗ ⟨ze, λ, n*1*, ..., nk⟩*. A priority-

*A*

multicounter-automaton can be changed in such a way that it has only one ac- cepting state *ze* and that all counters are empty while accepting. Thus, *L*(*A*) =

*{w | ⟨z*0*, w,* 0*, ...,* 0*⟩ | ∗ ⟨ze, λ,* 0*, ...,* 0*⟩}*.

*A*

Using Theorem [5.1](#_bookmark27), we show that the emptiness problem of the accepted lan- guage is decidable for priority-multicounter-automata. The same holds for the halt- ing problem by constructing an automaton which contains its input in the states.

**Theorem 6.1** *The emptiness problem for priority-multicounter-automata is decid- able.*

**Proof.** Given *A* we construct a Petri net (*P, T, W, I, m*0*, m*1) with the places *P* :=

*{*1*...k}∪ Z*, the transitions *T* = *δ*, the weights *W* with

*W* (*z,* ((*z', a, j*)*,* (*z'',V* ))) := 1 if *z* = *z'* else := 0;

*W* (((*z', a, j*)*,* (*z'',V* ))*, z*) := 1 if *z* = *z''* else := 0;

*W* (*i,* ((*z', a, j*)*,* (*z'',V* ))) := 1 if *V* (*i*) = *−*1 else := 0; and

*W* (((*z', a, j*)*,* (*z'',V* ))*, i*) := 1 if *V* (*i*) = 1 else := 0;

the inhibitor arcs *I* := *{*(*i,* ((*z', a, j*)*,* (*z'',V* ))) *| i ≤ j}*, the start marking **m**0 :=

*{z*0 *'→* 1*}*, and the end marking **m**1 := *{ze '→* 1*}* which is reachable from **m**0 if and only if *L*(*A*) */*= *∅*. According to Corollary [4.2](#_bookmark18) with **g**(*i*) = *i* for *i ≤ k* and **g**(*z*) = *k* +1 for *z ∈ Z*, this is decidable.

Consequences of this result for the decidability of the synchronizability and maximality of semi-trace languages are formulated in [[16](#_bookmark46)]. Further consequences are described in [[6](#_bookmark38)]: Every unary language is recursive if it is generated by a graph- controlled grammar with an arbitrary number of nonterminal symbols but only one of the nonterminal symbols being allowed to be used in the appearance checking mode. Theorem [6.1](#_bookmark29) implies the optimality the results in [[6](#_bookmark38)] proving the compu- tational completeness of graph-controlled grammars, programmed grammars, and

matrix grammars with a certain nonterminal complexity.

The classes *k-PMC* of languages accepted by a priority-multicounter-automaton with *k >* 0 counters (and also their union) are incomparable to the class *LIN* of linear languages and it holds *(k-1)-PMC* Ç *k-PMC*. This is because

*{an*1 *ban*2 *...bank*+1 $*ank*+1 *b...an*2 *ban*1 *| 6i ≤ k* +1 *ni ∈* N*} /∈ k−PMC.*

This can be shown by constructing *T* fulfilling property *7* and, then, by using Lemma [4.3](#_bookmark24) to find two different words in the language where the automaton has the same configuration reading $. With the same argument, this also holds for the classes *k-BLIND* and *k-PBLIND* in [[7](#_bookmark39)]. Furthermore, *{*(*anb*)*m | n, m ∈* N*}* cannot be accepted by a priority-multicounter-automaton (Theorem 3.2 in [[16](#_bookmark46)]).

# Restricted Priority- Multipushdown- Automata

We define a priority-multipushdown-automaton by a different treatment of one of the two pushdown symbols according to an order of the pushdown stores in the following way: let the pushdown alphabet be *{*0*,* 1*}*. A 0 can be pushed to and popped from every pushdown store independently, but a 1 can only be pushed to or popped from a pushdown store if all pushdown stores with a lower order are empty. Furthermore, the restriction requires that if a 1 is popped from a pushdown store, then a 1 cannot be pushed anymore to this store until it is empty.

**Theorem 7.1** *The emptiness problem for restricted priority-multipushdown-auto- mata is decidable.*

This generalizes the result in [[10](#_bookmark40)] that *LIN* %*D' ∗* (the class of languages gener- ated by linear grammar and deletion of semi Dyck words) is recursive. We conjecture that decidability still holds in the unrestricted case but, even in the special case of a pushdown automaton with additional weak counters (without zero-test), this is still an open problem.

1

# Acknowledgement

I thank Klaus-Jo¨rn Lange and Andreas Krebs, for many helpful discussions on this version and also Volker Diekert and Henning Fernau for many helpful discussions on previous versions.

# References

1. Arnon Avron. Transitive closure and the mechanization of mathematics. In F. Kamareddine, editor,

*Thirty Five Years of Automating Mathematics*, pages 149–171. Kluwer Academic Publishers, 2003.

1. L. A. Cherkasova and V. E. Kotov. Structured nets. In J. Gruska and M. Chytil, editors, *Proceedings of the 6th MFCS*, number 118 in LNCS, pages 242–251. Springer, 1986.
2. L. E. Dickson. Finiteness of the odd perfect and primitive abundant numbers with n distinct prime factors. *Amer. J. Math.*, 35:413–422, 1913.
3. N. Dershowitz and Z. Manna. Proving Termination with Multiset Orderings. *Comm. ACM*, 22(8):465– 476, 1979.
4. S. Eilenberg and M. P. Schu¨tzenberger. Rational sets in commutative monoids. *Journal of Algebra*, 13:173–191, 1969.
5. H. Fernau, R. Freund, M. Oswald, and K. Reinhardt. Refining the nonterminal complexity of graph- controlled grammars. In *Proceedings of the DCFS*, pages 110–121, 2005.
6. S. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoret. Comput.* *Sci.*, 7:311–324, 1978.
7. S. Ginsburg and E. H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific J. Math.*, 16:285–296, 1965.
8. N. Immerman. Languages that capture complexity classes. *SIAM J. of Computing*, 16:4:760–778, 1987.
9. M. Jantzen, M. Kudlek, K.-J. Lange, and H. Petersen. Dyck1-reductions of context-free languages. In

*Computers and Artificial Intelligence*, volume 9, pages 228–236, 1990.

1. H. Kleine Bu¨ning, T. Lettmann, and E. W. Mayr. Projections of vector addition system reachability sets are semilinear. *Theoret. Comput. Sci.*, 64:343–350, 1989.
2. S. R. Kosaraju. Decidability of reachability in vector addition systems. In *Proceedings 14th Ann. ACM* *STOC*, pages 267–281, 1984.
3. J.L Lambert. A structure to decide reachability in petri nets. *Theoretical Computer Science*, 99:79–104, 1992.
4. E. Mayr. An algorithm for the general Petri net reachability problem. *Siam J. Comput.*, 13:441–459, 1984.
5. M. L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1971.
6. K. Reinhardt. *Priorit¨atsz¨ahlerautomaten und die Synchronisation von Halbspursprachen*. Dissertation, Institut fu¨r Informatik, Universit¨at Stuttgart, 1994.
7. K. Reinhardt. *Counting as Method, Model and Task in Theoretical Computer Science*. habilitation- thesis, University of Tu¨bingen, 2005.