COMPUTATION OF GENERATIVE FAMILIES OF POSITIVE SEMI-FLOWS IN TWO TYPES OF COLOURED NETS

J.M. COUVREUR S. HADDAD J.F. PEYRE

Universite Paris VI -C.N.R.S. MASI et C3
4 Place Jussieu 75252 Paris Cedex 05

ABSTRACT

It is well known that a generative family of positive flows provides a more accurate information than a generative family of ordinary ones. For instance with the help of positive flows one can decide the structural boundness of the nets and detect the structural implicit places. Up to now, no computation of positive flows has been developed for coloured nets. In this paper, we present a computation of positive flows for two basic families of coloured nets: unary regular nets and unary Predicate/Transition nets. First of all, we show that these two computations are based on the resolution of the parametrized equation $A.X_1=A.X_2=...=A.X_n$ where A is a matrix and X_i, the unknowns are vectors. Thus an algorithm is presented to solve this equation and at last we show how this algorithm can be used to compute the generative family of semi-flows in the unary regular nets and unary Predicate/Transition nets.

KEYWORDS

Coloured nets, structural analysis, positive flows computation, Farkas algorithm

CONTENTS

INTRODUCTION

In Petri net theory. the computation of a set of integer vectors - called flows - or a set of nonnegative integer vectors - called semi-flows - is one of the key point for the analysis of systems [Mem83], [Si185], [Tre86]. So, as abbreviations of Petri nets - coloured nets [Jen82] and predicate transition nets [Gen81], [Lau85] - were introduced in order to model complex systems, many researchers have contributed to extend the main results of the Petri net theory and in particular the flow and semi-flow calculi.

These researches have provided, for the flows computation. a general algorithm [Cou89] in case where the size of colour domains are fixed -non parametrized algorithm - and many algorithms [Had86], [Cou90], [SiI85] working on subclasses of coloured nets which allow to do not fix the size of colour domains - parametrized algorithms -.Unfortunately the Algebra technics on which these results are based can no more be applied in the case of semi-flows calculus because $\boldsymbol{Q}^+$ - instead of $\boldsymbol{Q}$ - is not a ring. Thus only few heuristics have been proposed for the computation of semi-flows [Vau84], [SiI85], [Tre86].

In this paper we present the parametrized computation of a generative family of semi-flows for two categories of parametrized coloured nets: the unary regular nets (M-Rr nets)[Had86] and the unary predicate/transitions nets (U-Pr/T nets) [Gen81], [Vau84].

Section 1 of this paper reviews the two models of coloured nets on which our algorithm works and gives two examples of modelization with these models.

In section 2 we set precisely the problem of finding positive flows and its parametrization; we show that this parametrization can be reduced to the parametrized equation $D.X_1=D.X_2=...=D.X_n$ where D is the differential matrix of the net [Had86]. notion that is also extended to unary Predicate/Transition nets.

In section 3 we solve this last equation; i.e. we show that the set of solutions -for all $n \geq 2$ - can be generated by a finite - and calculable -set of integer vectors.

In section 4 we use this result to compute a generative family of semi-flows in unary regular nets and in unary predicate/transitions nets. We apply our algorithm on the two examples proposed in the first part.

I SUBCLASSES OF COLOURED NETS

This section will briefly review the most basic definitions of coloured net - multi-set, linear application, coloured net, incidence matrix, firing rule - and the definition of two subclasses of coloured net: the unary regular net and the unary predicate transitions nets (U-Pr/T nets). The reader who wants more details about these definitions could refer to the original papers [Jen86], [Had86], [Gen81].

I-1 Coloured nets

Definition 1.1: A multi-set, over a finite non-empty set A, is a mapping $a \in [A \rightarrow \mathbf{N}]$ where **N** is the set of all non negative integers.

Intuitively, a multi-set is a set which can contain multiple occurrences of the same element. Each multi-set a over A is represented as a formal sum :

$a = \sum a(x).x$ for all x in A, and where the nonnegative integer a(x) denotes the number of occurrences of the element *x* in the multi-set a.

We will denote by **Bag(A)** the set of multi-sets over A.

The relation order on **Bag(A)** is the natural extension of the relation order on **N**.

Definition 1.2: Let $a=\sum a(x).x$ and $b=\sum b(x)$ be two elements of Bag(A). We say that a is greater or equal than b -denoted $a \geq b$ - iff $\forall x \in A$, $a(x) \geq b(x)$.

Definition 1.3: A mapping $f \in [Bag(A) \rightarrow Bag(B)]$ is a linear application iff: $\forall a,a' \in Bag(A)$ $f(a+a') = f(a) + f(a')$ and $\forall \lambda \in \mathbf{N}$, $f(\lambda a) = \lambda f(a)$.

A linear application can be defined as the unique linear extension of a mapping in $[A \rightarrow Bag(B)]$. We could also define the $A \times B$ matrix of the linear application.

Definition 1.4: A *coloured net* is a 6-tuple *CPN* = $<P,T,C,W+,W-,M_0>$ where :

-P is the non-empty set of places.

-T is the non-empty set of transitions disjoint from P.

-C is the colour function: $C : P \cup T \rightarrow \Omega$, where Ω is a set of finite non-empty sets.

$\forall s \in P \cup T$, C(s) is the colour set (or domain) of s.

-W^+ (W^-) is the post (pre) incidence matrix defined from $P \times T$. $W^+(p,t)$ and $W^-(p,t)$ are linear applications of $[Bag(C(t)) \rightarrow Bag((C(p))]$.

-The initial marking $M_0(p)$ of the place p is an item of Bag(C(p)).

-The incidence matrix W of a coloured net is defined by $W = W^+ - W^-$.

Definition 1.6:

-A transition t is enabled for a marking M and a colour $c_t \in C(t)$ *iff:*

$\forall p \in P$, $M(p) \geq W^-(p,t)(c_t)$

-The firing of t for a marking M and a colour $c_t \in C(t)$ gives a new marking M' defined by:

$\forall p \in P$, $M'(p) = M(p) + W(p,t)(c_t)$

The unary regular nets [Had86] or the unary predicate/transition nets are characterized in the class of the coloured nets by a structuring of the domains and the functions. These two models are widely used in practice.

1.2 Unary regular nets

To each unary regular net is associated a colour domain, or class, which contains a certain number of colour, or objets. This number represents the parameter of the model.

The colour domain of each place and each transition is this class. The colour functions are constructed by addition and composition of two basic functions X et S, which allow to express the free evolution of an objet for the function X, and a global synchronization or the diffusion of all the objets for the function S.

Definition 1.7: Let E be a set then the functions <X> and <S> are defined by:

-<X> is the function from E to Bag(E) such as

<X>(x) = <x> for each x in E

-<S> is the function from E to Bag(E) such as

<S>(x) = $\sum$<e>, sum on E, for each x in E

Definition 1.8: An unary regular net is a 5-tuple $R = <P,T,C,W^+,W^->$ where

-P is the set of places

-T is the set of transitions, disjoint from P

-C is the domain colour noted C = [1..n], with n the parameter of the net

-W^+ and W^- are the incidence matrices where $W^+(p,t)$ and $W^-(p,t)$ are called colour functions

The incidence matrix W of a unary regular net is defined by $W = W^+ - W^-$. and we note for all p in P and all t in T, $W(p,t) = d_{p,t}.X + b_{p,t}.S$.

The firing rule is identical to the one of coloured nets.

We give an example of modelization with unary regular net. This example, defined in [Had86]. models a database.

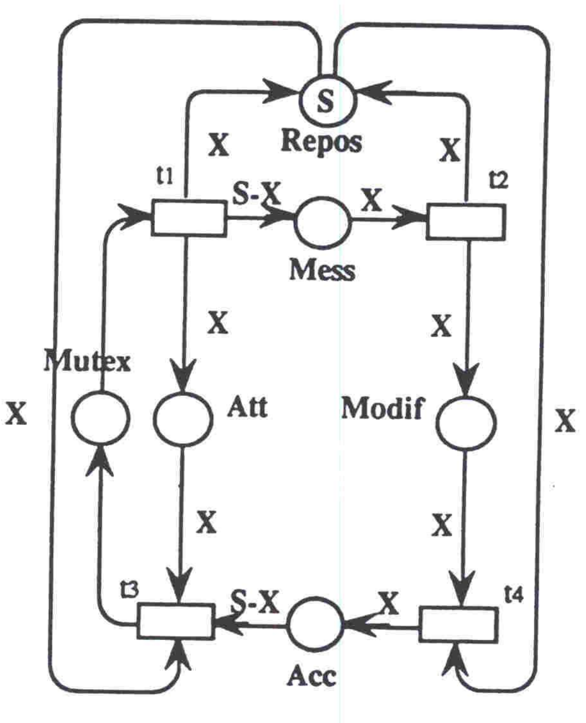


Figure 1

I-3 Unary predicate/transitions nets

We give the original definition of an unary predicate/transition net [Vau84].

Definition 1.9: An Unary-Predicate/Transition net is a 7-uple $R=<P,T,A,Y,W^+,W^-,Dom>$ where:

- P and T are two disjoint finite sets of places and transitions respectively.
- A and Y are two disjoints finite non empty sets.
 We shall call respectively colour an element of A, variable an element of Y, label an element of Bag(A+Y) and interpretation a function from Y to A. Σ will denote the set of all interpretations.
- W^+ and W^- are two functions from $P \times T$ to Bag(A+Y), respectively called forward incidence function and backward incidence function.
-Dom is a function defined on T such that, for each transition t, Dom(t) is a set of interpretation called the domain of the transition and denoted by D_t.

We note $Y = \{Y_1, Y_2,..., Y_{|Y|})$ the variables of the nets, and $A = \{a_1,a_2,...,a_q,1,2,...,n\}$ the domain colour with $a_1,a_2,\dots,a_q$ the constants of the net – i.e. the particular colours which may appear on the valuation of the arcs.

The incidence matrix W is defined by $W = W^+ - W^-$ and we note for all p in P and all t in T

$$W(p.t) = \sum_{y=l}^{|Y|} \delta_y^{p,t}.Y_y + \sum_{i=l}^{q} \alpha_i^{p,t}.a_i \text{ (a formal sum of constants and variables)}$$

The firing role is identical to the one of coloured nets.

We give an example of unary predicate/transition net which does not model something special but will allows us to illustrate the positive flows computation on this type of coloured net.

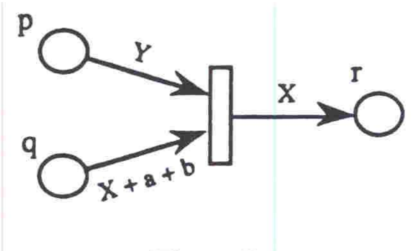


Figure 2

II SEMI-FLOWS IN COLOURED NETS

Semi-flows computation is an analysis tool much more important in coloured nets than in ordinary Petri nets. It enables to verify behavioural constraints without unfolding all or a part of the reachability graph which for a coloured net would have an untractable size even by a computer. It gives also informations on the evolution of the tokens and is finally the efficient auxiliary of others proof methods like, for example, the reduction theory.

Many categories of semi-flows have been introduced in [Lau85], [Vau85], but most of them cannot be placed in an appropriate algebraic context. Thus the authors who looked into the semi-flows or flows calculus [Vau86], [Silv85], [Had86], [Cou86] have taken the most natural definition of a semi-flows: a semi-flow of a coloured net corresponds to a set of semi-flows in the Petri net obtained by unfolding the initial coloured net.

Thus, given a coloured net, a solution for computing its semi-flows can be to unfold this net and then to compute the semi-flows with the help of classical methods. However. such a solution is expensive in time and in space - because of the sire of th unfolded net - and provides invariants whose meaning is not always clear. Furthermore. when the size of domains colour are not fixed - case of parametrized nets - it becomes impossible to unfold the net and then to compute semi-flows. Thus, it is necessary to find alternative methods.

II-1 PRELIMINARIES

II-1.1 Notations

Let p be the size of the set P of places. Semi-flows of a Petri net are vectors of $(\boldsymbol{Q}^+)^p$, where $\boldsymbol{Q}^+$ denote the nonnegative rationals and semi-flows of a unary regular net or of a U-P/T net are vectors of $((\boldsymbol{Q}^+)^p)^n$, where n is an integer greater than 1 representing the parameter of the model.

We introduce the following notations:

- ***N*** denotes the set of non negative integers
- ***Q***$^+$ denotes the set of non negative rationals
- We note $E = (\boldsymbol{Q}^+)^p$ and $E^n = ((\boldsymbol{Q}^+)^P)^n$ where *p* and *n* are non negative integers
- So $E^n = \oplus E_i$. for all i in [1..n]
- The image of a vector e of E by the canonical bijection from E to E_i is noted *e(i)*
- A vector e of E^n has a unique decomposition $e = \sum e_i(i)$ or $e = <e_1, e_2, \ldots, e_n>$ with e_i in E
- *P*(E) denotes the powerset of E.

II-1.2 Definitions

We recall now some basic definitions of Petri nets.

Definition 2.1: Let R be a Petri net and W its incidence matrix. A vector $f \in E$ is a *semi-flow* iff f is a solution of the equation $f^t.W=0$.

Definition 2.2 : Let V be a vector indexed by E, the *support* of V noted Supp(V) or [[V]] is the subset of E defined by $[[V]] = \{e \in E \mid V_e \neq 0\}$.

Definition 2.3 : Let F be a set of vectors of E. A vector g of F is minimal with respect to the support notion *iff* :

$$\forall f \in F \backslash \{g\}, [[f]] \not\subset [[g]]$$

Example:

Let F={ f=(2,0,0,3), f_1=(1,1,0,2), f_2=(l,l,3,0)} a set of vector of Q^4; then [[f]] can be viewed as the boolean vector (1,0,0,1) and f is of minimal support in F.

Definition 2.4: Let W be the incidence matrix of a Petri net R. A set $\{f_1,\ldots,f_k\}$ of vectors of E is a ***generative family*** of semi-flows of R iff:

- $\forall$ i, $f_i^t.W = 0$
- $\forall f \in E$ $f \neq 0$ with $f^t.W=0$, $\exists \lambda_1,\ldots,\lambda_k \in Q^+$ with $f= \lambda_1.f_1+\ldots+ \lambda_k.f_k$

It is also possible to characterize a generative family with the help of the notion of support [Mem83], [Cam68].

Characterization 1: Let W be the incidence matrix of a Petri net R. A set $\{f_1,\ldots,f_k\}$ of vectors of E is a generative family of semi-flows of R *iff:*

- $\forall$ i, $f_i^t.W = 0$
- $\forall f \in E$ $f \neq 0$ with $f^t.W=0$, $\exists f_i$ e F such that $[[f]] \supset [[f_i]]$.

This characterization and the definition 3.4 provide the Farkas algorithm [Far02] which allows to compute the minimal generative family of semi-flows in a Petri net and which is studied in [Mem83] or more recently in [Col89]. This last paper gives heuristics providing an efficient programming of the Farkas algorithm.

We give now the definition of semi-flows in unary regular nets or in unary predicate/transition nets.

Definition 2.5: Let R be a M-Rr or a U-P/T net, with n the parameter of the colour domain. A positive flow is a symbolic expression such that for each n this expression gives a vector in E^n which is a positive flow for the net unfolded for this particular n.

II-2 A FIRST STEP INTO THE RESOLUTION OF THE PROBLEM

Computation of semi-flows needs to solve the equation W.X = 0, where W is the incidence matrix of the unfolded net. In unary regular net or in U-P/T net this matrix has p.n columns and t.n lines, where p and t are respectively the places and transitions number of the net, with n the parameter corresponding to the size of the domain colours.

Also we have to solve a system parametrized in two ways: the equations of this system are parametrized – the solutions belong to E^n -, and the number of equations depends on n - there are t.n equations-.

We shall see in this section, on specifying successively the incidence matrix of a M-Rr net and of a U-P/T net, that we can in the two cases reduce our problem to a system with only two equations including the following one: $D.X_1=D.X_2=...=D.X_n$ where D is a P×T matrix depending on the net

II-2.1 Case of unary regular nets.

Let R=< P, T ,C, W^+, W^- > be a unary regular net. The incidence matrix of the unfolding net is defined by:

$$\forall p\in P, \forall t \in T, \forall c,c' \in [1..n], W((p,c), (t,c')) = W(p,t)(c,c').$$

Thus if we denote W(p,t)(c,c') = $d_{p,t}.X + b_{p,t}.S$ with the definitions of part I-2.1, we have:

$$\forall\ p \in P, \forall\ t \in T, \forall\ c,c' \in [1..n]$$
$$W((p,c), (t,c')) = d_{p,t}+ b_{p,t} \qquad \text{if } c = c'$$
$$W((p,c), (t,c')) = b_{p,t} \qquad \text{if } c \neq c'$$

If we denote now by *D* the P×T matrix defined by D(p,t) = $d_{p,t}$ for all (p,t) in PxT and by *B* the PxT matrix defined by B(p,t) = $b_{p,t}$ for all (p,t) in P×T, the incidence matrix of the unfolded net can be written as :

$$W_n = \begin{array}{|ccccc|l} P_1 & P_2 & & P_n & & \\ D+B & B & \dots & B & & T_1 \\ B & D+B & \dots & B & & T_2 \\ B & B & \dots & B & & \\ \dots & \dots & \dots & \dots & & \\ B & B & \dots & D+B & & T_n \end{array}$$

where P_i denotes the set of places for the colour c=i and where T_i denotes the set of transitions for the colour c=i.

Thus a vector X = $<X_1,...,X_n>$ in E^n - with notations of part III-1.1 - is a semi-flow if and only if this vector verifies the system:

$$(1)\ \forall\ i \in\ [1..n],\ D.X_i+ \sum_{j=1}^{n} B.X_j = 0$$

Let us examine the system (2):

$$(2)\quad \begin{cases} (D+nB). \sum_{j=1}^{n} X_j = 0 \\ D.X_1 = D.X_2 = ...= D.X_n \end{cases}$$

We have the following proposition.

Proposition 1: Systems (1) and (2) are equivalent.

Proof: The system (1) implies the system (2): i.e. for each X solution of (1) then X is solution of (2).

$$\forall\, i,j \in [1..n]^2,\ (D.X_i + \sum_{k=1}^{n} B.X_k) - (D.X_j + \sum_{k=1}^{n} B.X_k) = 0$$

so it comes $\forall\, i,j \in [1..n]^2$, $D.X_i = D.X_j$ which can also be written $D.X_1 = D.X_2 = ...= D.X_n$

$$\sum_{i=1}^{n}(D.X_i + \sum_{j=1}^{n} B.X_j) = 0 \text{ so, } D.(\sum_{j=1}^{n} X_j) + nB.(\sum_{j=1}^{n} X_j) = 0$$

The system (2) implies the system (1):

$(D+n.B)\sum_{j=1}^{n} X_j = 0$ and $\forall\, i,j \in [1..n]^2$, $D.X_i = D.X_j$ so we have:

$$\forall\, i \in [1..n],\ n.D.X_i + n.B.\sum_{j=1}^{n} X_j = 0$$

◊◊◊

Thus computing a generative family of semi-flows for a unary regular net is equivalent to solve the system (2).

11-2.2 Case of unary predicate transition nets

Let R = $<P,T,A,Y,W^+,W^-,Dom>$ be a U-P/T net and let $W = W^+ - W^-$ be the incidence matrix of this net.

As defined in section II-3 we note $Y = \{Y_1,Y_2,...,Y_{|Y|}\}$ the variables of the nets, and $A = \{a_1,a_2,...,a_q,1,2,...,n\}$ the domain colour with $a_1,a_2,...,a_q$ the constants of the nets; there is also n+q colours, where q is the number of constants of the net and n the parameter of the net.

We note W the incidence matrix with $W(p,t) = \sum_{y=1}^{|Y|}\delta_y^{p,t}.Y_y + \sum_{i=1}^{q} \alpha_i^{p,t}.a_i$

The incidence matrix Wd_n of the unfolded net is defined by:

$$\forall\, p \in P, \forall\, t \in T, \forall\, a \in A, \forall\, \sigma \in D_t, Wd_n((p,a),(t,\sigma))=\sigma(W(p,t))(a)$$

Let now *D* be the matrix defined by: a column for each place of the net, and a row for each transition and each variable with :

$$\forall\, p \in P, \forall\, t \in T, \forall\, y \in Y, D(p,t_y)= \delta_y^{p,t}$$

Let i_0 be a particular colour distinct of the constants – $i_0 \in A\backslash\{a_1,a_2,...,a_q\}$ -, and let $H_{i0},H_{a1},\dots,H_{aq}$, be the PxT matrices defined by:

-$\forall\, p \in P, \forall\, t \in T, H_{i0}(p,t)=\sum_{y=1}^{|Y|} \delta_y^{p,t}$

-$\forall\, i \in [1..q], \forall\, p \in P, \forall\, t \in T, H_{ai}(p,t) = \alpha_i^{p,t}$

Remark: All these matrices D, $H_{i0},H_{a1},\dots,H_{aq}$ are finite because P, T, Y and $\{ a_1,a_2,...,a_q \}$ are finite sets.

Let finally $W'd_n$ be the following bloc matrix:

$$W'd_n = \begin{array}{c|cccccc|c} & P_{i0} & P_{a1} & & P_{aq} & & P_n & \\ & H_{i0} & H_{a1} & \dots & H_{aq} & \dots & 0 & T_{i0} \\ & D & -D & \dots & 0 & \dots & 0 & T_{a1} \\ & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & D & 0 & \dots & -D & \dots & 0 & T_{aq} \\ & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & D & 0 & \dots & 0 & \dots & -D & T_n \end{array}$$

We have the following proposition which is the interesting result of this part.

Proposition 2 : For each n, Wd_n and $W'd_n$ are equivalent for the calculus of semi-flows;
i.e. : $\forall n \geq 0$, $\forall X \in E^{n+k}$, $Wd_n.X = 0$ *iff* $W'd_n.X = 0$.

Sketch of Proof:

The proof lies on the fact that $W'd_n$ is obtained from Wd_n by linear combinations on the rows and on the fact that if we note $\sigma_{y=a} \equiv \{ \sigma \in \cup D_t \mid \sigma(y) = a)$, we have:

$\forall p \in P, \forall t \in T, \forall y \in Y, \forall a \in A/\{i_0\}, \forall \sigma \in \sigma_{y=a},$

Let $\sigma' \in D_t$ with $\sigma'(y) = i_0$ and $\sigma'(x) = \sigma(x)$ for all $x \in Y$, $x \neq y$ we have:

- $Wd_n((p,a),(t,\sigma)) - Wd_n((p,a),(t,\sigma')) = \delta_y^{p,t}$
- $Wd_n((p,i_0),(t,\sigma)) - Wd_n((P,i_0),(t,\sigma')) = -\delta_y^{p,t}$
- $Wd_n((p,b),(t,\sigma)) - Wd_n((p,b),(t,\sigma')) = 0$ for all b in A with $b \neq a$ and $b \neq i_0$

For a complete proof please refer to [Pey90].

So we have translated the problem of computing semi-flows of an U-P/T net into the problem of finding solutions of the equation $W'd_n.X = 0$ which can also be written:

$$H_{i0} . X_{i0} + H_{al} . X_{al} + \ldots + H_{aq} . X_{aq} = 0$$

(3) and

$$D.X_{al} = D.X_{a2} = \ldots = D.X_{aq} = D.X_1 = \ldots = DX_{i0} = \ldots = DX_n$$

Computing the generative family of semi-flows for a U-P/T net is equivalent to compute the generative family of solutions of the system (3).

We find again in this system the equation $D.X_1 = D.X_2 = \ldots = D.X_n$. Let us see now how to solve it.

III RESOLUTION OF THE PARAMETRIZED EQUATION $A.X_l = \ldots = A.X_n$

Our purpose is now to propose an algorithm which computes a generative family of the equation $A.X_1 = \ldots = A.X_n$ for any finite matrix A. We first define the notion of pseudo-generative family. Then, in order to compare pseudo-generative families, we extend the support notion to families of vectors, and we give then the algorithm 2 which is a solution to our problem.

From up to now, we denote by A a matrix composed by *t* rows and *p* columns, by b a vector of $\boldsymbol{Q}^t$ and we note *(i)* the equation $A.X_1 = A.X_2 = \ldots = A.X_n$.

Definition 3.1: Let F = $\{V_1,...,V_m\}$ be a family of vectors of E solutions of the equation A.X=b. F is a *pseudo-generative family* of A.X=b *iff:*

$\forall$ V $\in$ E , V$\neq$0 with A.V=b then $\exists$ $V_i \in$ F with [[V]] $\supset$ [[V_i]].

We note Sol(A,b) a pseudo-generative family of A.X=b.

Remark: According to characterization 1, Sol(A,0) is a generative family of the equation A.X=0.

We propose now an algorithm which, given a matrix A and a vector b, computes a pseudo-generative family of A.X=b.

Algorithm 1:

```
I) Compute - with Farkas - a generative family F = $\{(X, \lambda)_{X \in E, \lambda \in \mathbf{N}} \mid [A\ -b].(X , \lambda) = 0\}$
2) Remove from this family the solutions such that the second component $\lambda$ is null.
3) For each solution divide the first component X by the second component $\lambda$, to normalize the family;
then eliminate the component $\lambda$.
```

The family F of vectors of E obtained at the step 3 is a pseudo-generative family of the equation A.X=b.

Proof of the algorithm:

-$\forall$ V $\in$ F, A.V = b by construction.

-$\forall$ V $\in$ E, V $\neq$ 0 and A.V = b, because the family F obtained at the first step is a generative family, $\exists$ { ((V_i,λ_i), c_i) } such that (V,1) = $\sum c_i$.(V_i,λ_i) with $\forall$ i, [[(V,1)]] $\supset$ [[(V_i,λ_i)]]. Obviously, $\exists$ j such that $\lambda_j \neq 0$; thus $(1/\lambda_j).V_j \in$ F and [[V]] $\supset$ [[$(1/\lambda_j).V_j$]].

◊◊◊

Definition 3.2:

If b$\neq$0, we note $Sol^n(A,b)=\{ \sum V_i(i) \in E^n \mid \forall\ i, V_i \in Sol(A,b)\}$.

$Soln(A,0)=\{ V(i) \in E^n \mid V \in Sol(A,0)\}$.

Remark:

We use also the following notations to denote the vectors of Sol(A,b) and of $Sol^n(A,b)$:

$Sol(A,b) = \{V_1^b,\ldots, V_{mb}^b\}$

$$Sol^n(A,b)= \{X \in E^n \mid X = \sum_{v_k^b \in Sol(A,b)} \sum_{i \in C_k} V_k^b\ (i) \text{ with } \{C_k\} \text{ a partition of } [l,n]\}$$

or

$$Sol^n(A,b) = \{ X \in E^n \mid X =\sum_{i=1}^{n} V_{\sigma\text{-}1(i)}^b\ (i)\ (i) \text{ with } \sigma \text{ an application from } [1,n] \text{ to } [1,mb]\}$$

Example :

If Sol(A,b) = $\{V_1, V_2\}$, then:

$Sol^3(A,b) = \{<V_1,V_1,V_1>, <V_1,V_1,V_2>,\ldots,<V_2,V_1,V_2>,...,<V_2,V_2,V_2>\}$.

If we remark now that, for any b, $Sol^n(A,b)$ gives a set of solution of the equation *(i),* one can think is sufficient to solve (i) for n=2, i.e. to solve A.X=A.Y. We are going to see on an example, that in general, a generative family of A.X=A.Y does not give a generative family of (i) for all n.

The following algorithm computes a generative family of A.X=A.Y.

Algorithm 2 : A.X=A.Y

1) Compute - with Farkas – a generative family Fg= { (X,Y) / [A -A].(X,Y)=0}
2) S = { b | b= A.X for some (X,Y) $\in$ Fg }

F = $\cup_{b\in S\cup\{0\}}$ $Sol^2(A,b)$ is a generative family of A.X=A.Y.

Proof:

It is clear that F contains a generative family of A.X=A.Y

◊◊◊

Fact:

The family computed by the algorithm 2 does not provide a generative family of the equation $A.X_1 = A.X_2 = ...= A.X_n$ for all $n \geq 2$.

Let be A the matrix:

$$A = \left|\begin{array}{ccccccc} 1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & 1 \end{array}\right|$$

The set of b_i computed by the algorithm 2 is (with on the right of each b_i the set $Sol(A,b_i)$).
((_ 1 3_ _ _ _) denotes the vector (0 1 3 0 0 0 0)).

b_0 = (0 -2 0 0)	X = (_ 1 1 _ _ _ _)	b_l = (1 -1 -1 1)	X = (_ _ 1 _ _ _ _)
	X = (_ _ _ _1 _ 1)		X = (1 _ _ _ 1 1 2)
b_2= (1 -1 1 -1)	X = (_ _ _ _1_ _)	b_3 = (-1 -1 -1 -1)	X = (_ _ _ 1_ _ _)
	X = (1 2 1 _ _ 1 _)		X = (_ 1 _ _ _ 1 1)
b_4 = (0 -2 0 -2)	X = (_ _ _ 1 1 _ _ l)	b_5 = (0 -2 -2 0)	X = (_ _ 1 1 _ _ _)
	X = (_ 1 _ _ 1 1 1)		X = (_ _ _ _ 1 1 2)
	X = (_ 2 1 _ _ 1 _)		X = (_ 1 1 _ _ 1 1)

Let the vector b_6 - with $Sol(A,b_6)$ = $\{X_1,X_2,X_3)$ - be defined by:

b_6=(1 -3 -1 -1)	X_l = (_ _ _ _ 2 1 2)
	X_2 = (_ _ 1 1 1 _ _)
	X_3 = (_ 2 2 _ _ l _)

This last vector is not computed by the algorithm 2 because none vector of $Sol^2(A,b_6)$ is minimal in support between the set ($\cup$ $Sol^2(A,b_i)$ $b_i \in$ [b_0,...,b_5] }.Nevertheless this vector b_6 gives for n=3 the vector of E^3 X=<X_1,X_2,X_3> which is minimal in the set { $\cup$ ($Sol^3(A,b_i)$ with $b_i \in$ {b_0,...,b_5}}. Thus, since we want a generative family, we have to compute this vector.

It is thus necessary to iterate this calculus; we want to say that after solving $A.X_l=A.X_2$ we have to solve $A.X_1=A.X_2=A.X_3$ and then $A.X_l=A.X_2=A.X_3=A.X_4$ and so on. Does such an algorithm end? That is the question.

In fact, the reason why we have to iterate the calculus implies that this iteration is bounded. If a vector b is missing, it is because this vector generates a minimal - in support - solution in E^n, and it cannot generate a minimal solution if for each b_i already computed the set Sol(A,b) does not contain a minimal vector in the set Sol(A,b_j). Also a vector b is missing *if and only if* the set Sol(A,b) is in a way *"minimal"* in the set {$\cup$Sol(A,b_i),b_i already computed). We are going to see that it cannot exist a infinite sequence of *"minimal"* sets and then, that the number of necessary iterations is bounded.

In order to clarify this notion of "minimal" vectors sets, we extend the notion of support.

Definition 3.4:

Supp : $\mathcal{P}(E) \rightarrow \mathcal{P}((\mathcal{P}(P))$
such that Supp(F) = { Supp(V) | V $\in$ F }; by extension we note [[F]] = Supp(F).

Definition 3.5:

Let F and F two sets of vectors of E.
We say that F is *minimal* in comparison with F - noted F < F' - *iff*
$\exists$ f $\in$F such that $\forall$ f' $\in$F [[f']] $\not\subset$ [[f]]

We have for the last example, Sol(A,b_6) < Sol(A,b_i) for each b_i and it is why we have to compute the vector b_6.

The following proposition shows that it cannot exist an infinite sequence of *"minimal"* sets.

Proposition 3.6 :

$\forall$ $\{F_i\}_{i \in \mathbf{N}}$ with $F_i \in \mathcal{P}(E)$, $\exists$ i_0,j_0 such that $F_{i0} < F_{j0}$.

Proof:

The sequence $\{[[F_i]]\}_{i \in \mathbf{N}}$ takes its values in $\mathcal{P}((\mathcal{P}(P))$. As $\mathcal{P}((\mathcal{P}(P))$ is a finite set, $\exists$ i_0,j_0 with $i_0<j_0$ such as [[F_{i0}]] = [[F_{j0}]]. Thus, by definition, we have $F_{i0} < F_{j0}$.
◊◊◊

We give now the algorithm which computes a generative family of the equation $A.X_l=...=A.X_n$.

Algorithm 3 :

```
1) S = {$b_0$ ,...,$b_k$} // result of the algorithm 2 on the matrix A (A.X = A.Y)
2) IncS := $\emptyset$
3) DO
   3.1) K = | S | // cardinality of S
   3.2) Compute - with Farkas - a smallest generative family F
              F={$(X,\lambda_1,...,\lambda_K) \in E \times (\mathbf{Q}^+)^K$ | [$-b_1 \ldots -b_K$ A]. $(\lambda_1,...,\lambda_K,X)=0$ }
   3.3) IncS:={b' | b'=A.X for some $(X,\lambda_1,...,\lambda_K) \in$ F, b'$\neq$0 and $\forall$ b$\in$S, Sol(A,b') < Sol(A,b)}
   3.4) S := S $\cup$ IncS
   WHILE IncS$\neq\emptyset$
4)S:=S$\cup${0}
```

Proposition 1.7 :

The set S computed by the last algorithm provides a generative family of the equation $A.X_l=\ldots=A.X_n$ for all $n\geq 2$

i.e. $\bigcup_{b\in S} Sol^n(A,b)$ is a generative family for all $n\geq 2$

Proof:

-Termination:

If this algorithm does not terminate, it builds with the instruction 3.4 a infinite sequence $\{b_i\}$ indexed by their insertion order in S. If we consider the family $\{Sol(A,b_i)\}$, the instruction 3.3 implies that this family does not satisfy the proposition 3,6. Hence there is a contradiction.

-Correctness:

We do not give the proof because of its technical nature but the reader might refer to [CHP90], ◊◊◊

IV APPLICATION TO THE SEMI-FLOWS COMPUTATION IN COLOURED NETS

We are going to see now how we use the results developed in the last part to compute a generative family of semi-flows in two types of coloured nets.

IV-1 Unary Predicates/Transitions nets

We recall that computing a generative family of semi-flows in a U-P/T net is equivalent to compute a generative family of the system (3) where D, H_{i0}, H_{al},..., H_{aq} are the matrices defined in part II-2.2,

$$(3)\quad \begin{array}{l} H_{i0} . X_{i0} + H_{al} . X_{al} +\ldots+H_{aq} . X_{aq} = 0 \\ \text{and} \\ D.X_{al}=D.X_{a2}=\ldots=D.X_{aq}=D.X_1=\ldots=DX_{i0}= \ldots=DX_n \end{array}$$

Also, we propose the following algorithm which computes a generative family of semi-flows in a U-P/T net, The principle is to solve the second equation by the algorithm developed in part III and then to report the solutions in the first equation. As the size of this equation is not dependent on n, it is just necessary to develop the solutions computed on the first q+1 components and to keep the rest of the components as a formal sum.

We recall notations used in part II.2.2:

- C = { $a_1,\ldots,a_q,1,\ldots,n$} the colour domain, q the number of constants
- S = ($b_0\ldots,b_m$) the result of the algorithm 3 on the matrix D ($b_0= 0$)
- m_{bj} denotes the size of Sol(D,b_j)
- Sol(D,b_j) = $\{V_1^{bj},\ldots,V_{mb}^{bj}\}$for each b_j in S

Algorithm 4:

```
1) Compute the set $S = \{b_j\}$ corresponding to the matrix D with the algorithm 3.
2) Express the solutions of $D.X_l=\ldots=D.X_n$ as a formal sum with only the first q+l components
   developed - for the constants and the particular colour $i_0$ in [l,n]- :
   - for each $b_j\neq 0$ in S, for each application $\sigma_{j,k}$ from [0,q] to $[1,m_{bj}]$ (0 for the colour $i_0$)
       form the symbolic vector $X_{j,k}$ in $E^n$ :
           $X_{j,k} = \sum_{c \in [l,q]} V_{\sigma j,k(c)}^{bj}(a_c) + V_{\sigma j,k(0)}^{bj}(i_0) + \sum_{i=1}^{m_{bj}} \sum_{c \in C_i^{j,k}} V_i^{bj}(c)$
           with $\{C_i^{j,k}\}_{i\in[l,mbj]}$ a partition of $[l,n]\backslash\{i_0\}$
   - for each $V_k^0$ in Sol(D,0), for each color c in $\{i_0\}\cup[l,q]$ form the vector $X_{0,kc}$ in $E^n$:
           $X_{0,kc} = V_k^0(a_c)$ if $c\neq i_0$ and $X_{0,kc} = V_k^0(io)$ if $c= i_0$ (only one component is non nul)
3) for each $X_{j,k}$ constructed at the step 2 do the projection $P(X_{j,k})$ on the first equation
   ( $X_{j,k}(i)$ denotes the i^th component of the vector $X_{j,k}$ ):
   $P(X_{j,k}) = H_{i0} . X_{j,k}(i_0) + H_{al}.X_{j,k}(a_1) +\ldots + H_{aq} . X_{j,k}(a_q)$
4) Solve by Farkas :
   $\sum_{j,k} \mu_{j,k}.P(X_{j,k}) = 0$
5) $S^* = \{<\mu_{j,k}>\}$ the computed family.
```

The following proposition makes the link between the family computed by the last algorithm and the solutions of the equation (3) i.e. the semi-flows of the U-P/T net:

Proposition 4.1:

Let F be the family composed by the vectors:

- $\forall$ i $\geq$q+1, $\forall$ X $\in$ Sol(D,0) the vector X(i).
- $\forall$ $<\mu_{j,k}> \in$ S*, the vector:

$$Z = \sum\mu_{j,k}.X_{j,k}$$

Then F is a generative family of semi-flows.

Proof

The proof lies on the fact that the number of constant is finite -independent of n -and that the Farkas algorithm computes a generative family.

For a complete proof, please refer to [CHP90].

◊◊◊

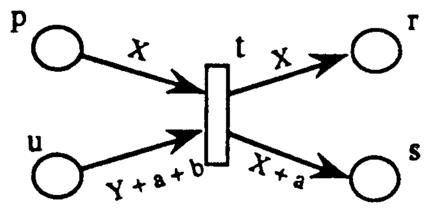
Remark

The family obtained is not necessary minimal.

Example:

Let be the net:

With D, H0, Ha, Hb the matrixes defined by:



$$D=\begin{array}{c} \\ \\ \end{array}\begin{array}{cccc} p & u & r & s \\ \end{array}$$

$$D=\begin{array}{ccccc} & p & u & r & s \\ tx & -1 & 0 & 1 & 1 \\ ty & 0 & -1 & 0 & 0 \end{array}$$

$$H0=\begin{array}{ccccc} & p & u & r & s \\ t & -1 & -1 & 1 & 1 \end{array}$$

$$Ha=\begin{array}{ccccc} & p & u & r & s \\ t & 0 & -1 & 0 & 1 \end{array}$$

$$Hb=\begin{array}{ccccc} & p & u & r & s \\ t & 0 & -1 & 0 & 0 \end{array}$$

The first step of the algorithm -algorithm 3 on the matrix D -computes the set defined by :
S = { 0=(0, 0), b_1 =(-1, 0), b_2=(0, -1), b_3=(1, 0) } with:

-Sol(D, 0) = {(p+r), (p+s)}
-Sol(D, b_1) = { (p) }
-Sol(D, b_2) = { (u) }
-Sol(D, b_3) = { (r), (s) }

We develop now the symbolic vectors associated to each b_i in S -step 2 -:

.b_1 :
Because there is only one vector in Sol(D,b_1), there is only one partition of [1,n] and one application σ from [0,2] to [1,1] : $\sigma(0)=\sigma(1)=\sigma(2)=1$.
Also the vector b_1 gives only one symbolic vector $X_{1,1}$ with:

$$X_{1,1} =\sum_{c\ e\ [1,2]} p(a_c) + p(i_0) +\sum_{c \in [1,n]\setminus\{i_0\}} p(c) = p(a) + p(b) + p(i_0) +\sum_{c \in [1,n]\setminus\{i_0\}} p(c)$$

.b_2:
For the same reasons the vector b_2 gives only one symbolic vector $X_{2,1}$ with:

$$X_{2,1} =\sum_{c\ e\ [1,2]} u(ac) + u(i_0) +\sum_{c \in [1,n]\setminus\{i_0\}} u(c) = u(a) + u(b) + u(i_0) +\sum_{c \in [1,n]\setminus\{i_0\}} u(c)$$

.b_3:
There are two vectors in Sol(A,b_3), two constants, a and b plus the particular color i_0. Also the vector b_3 gives eight symbolic vectors because there are eight applications from [0,2] to [1,2].

We note these vectors:

$$X_{3,rrr} = r(a) + r(b) + r(i_0) +\sum_{c_1 \in C_1} r(c) +\sum_{c_2 \in C_2} s(c) \text{ with } C_1 \oplus C_2 = [1,n]\setminus\{i_0\}$$

to

$$X_{3,sss} = s(a) + s(b) + s(io) +\sum_{c_1 \in C_1} r(c) +\sum_{c_2 \in C_2} s(c) \text{ with } C_1 \oplus C_2 = [1,n]\setminus\{i_0\}$$

$C_1 \oplus C_2$ denotes that {C_1 , C_2} is a partition of C

.$b_0 = 0$:

The vector 0 gives the six vectors: .

$X_{0,la} = (p+r)(a)$ $X_{0,2a} = (p+s)(a)$

$X_{0,lb} = (p+r)(b)$ $X_{0,2b} = (p+s)(b)$

$X_{0,li0} = (p+r)(i_0)$ $X_{0,2i0} = (p+s)(i_0)$

We have now to make the projection of these vectors on the first equation -step 3 -: Compute the sum: $P(X_{j,k}) = H_{i0}.X_{j,k}(i_0) + H_{a1}.X_{j,k}(a_l) +... + H_{aq}.X_{j,k}(a_q)$

We obtain:

.$P(X_{l,l})=-1$
.$P(X_{2,1}) = -3$
.$P(X_{3,rrr}) = P(X_{3,rrs}) = P(X_{3,rsr}) = P(X_{3,rss}) = 1$
.$P(X_{3,sss}) = P(X_{3,ssr}) = P(X_{3,srs}) = P(X_{3,srr}) = 1$
.$P(X_{0,la}> = P(X_{0,lb}) = P(X_{0,1i0}) =0$
.$P(X_{0,2b}) = P(X_{0,2i0}) =0$
.$P(X_{0,2a}) = 1$

As $P(X_{3,rrr})=P(X_{3,rrs})=P(X_{3,rsr})=P(X_{3,rss})$ and $P(X_{3,sss})=P(X_{3.ssr})=P(X_{3,srs})=P(X_{3,srr})$ and $P(X_{0,1a})=P(X_{0,lb})=P(X_{0,1i0}>$ and $P(X_{0,lb})=P(X_{0,li0})$ we rename the symbolic vectors :

$$X_{l,l} = \sum_{c \in [1,2]} p(a_c) + p(i_0) +\sum_{c \in [1,n]\setminus\{i_0\}} p(c) = p(a) + p(b) +\sum_{c \in [1,n]} p(c)$$

$$X_{2,1} = \sum_{c \in [1,2]} u(a_c) + u(i_0) +\sum_{c \in [1,n]\setminus\{i_0\}} u(c) = u(a) + u(b) +\sum_{c \in [1,n]} u(c)$$

$$X_{3,r(a)} = r(a) +\sum_{c \in C_1} r(c) +\sum_{c \in C_2} s(c) \text{ with } C_l \oplus C_2 = [l,n]\cup \{b\}$$

$$X_{3,s(a)} = s(a) +\sum_{c \in C_1} r(c) +\sum_{c \in C_2} s(c) \text{ with } C_l \oplus C_2 = [l,n]\cup\{b\}$$

$X_{0,l} = (p+r)(i)\ i \in \{a,b,i_0\}$

$X_{0,2a} = (p+s)(a)$

$X_{O,2bi0} = (p+s)(i)\ i \in \{b,i_0\}$

We execute now the fourth step of the algorithm: we solve the system:

$$\sum_{j,k} \mu_{j,k} . P(X_{j,k}) = 0$$

… and we obtain the generative family:

$\{ (\mu_{0,l}), (\mu_{0,2bi0}),$
$(\mu_{l,l} + \mu_{3,r(a)}),(\mu_{l,l} + \mu_{3,s(a)}), (\mu_{l,l} + \mu_{0,2a}),$
$(\mu_{2,1} + 3\mu_{3,r(a)}), (\mu_{2,1} + 3\mu_{3,s(a)}),(\mu_{2,1} + 3\mu_{0,2a})\}$

We just have now to develop the solutions with the help of proposition 5.1, and we obtain finally the generative family F composed by the vectors:

We develop first the vectors associated to the set Sol(D,0):

.Sol(D,0)
- $-(p + r)(i)$ for all $i \in [1,n]$ (i different of the constant)
- $-(p + s)(i)$ for all $i \in [1,n]$ (i different of the constant)

$.(\mu_{0,l})$

$-(p+r)(i)$ $i \in \{a,b,i_0\}$ whatever $i_0 \in [1,n]$

$. (\mu_{0,2bi0})$

$-(p+s)(i)$ $i \in \{b,i_0\}$ whatever $i_0 \in [1,n]$

$. (\mu_{l,l} + \mu_{3,r(a)})$

$-(p+r)(a) + \sum_{c \in C_l} (p+r)(c) + \sum_{c \in C_2} (p+s)(c)$ with $C_l \oplus C_2 = [l,n] \cup \{b\}$

$. (\mu_{l,l} + \mu_{3,s(a)})$

$-(p+s)(a) + \sum_{c \in C_l} (p+r)(c) + \sum_{c \in C_2} (p+s)(c)$ with $C_l \oplus C_2 = [l,n] \cup \{b\}$

$. (\mu_{l,l} + \mu_{0,2a})$

$-(p+s)(a) + \sum_{c \in [l,n] \cup \{a,b\}} p(c)$

$. (\mu_{2,1} + 3\mu_{3,r(a)})$

$-(u+3r)(a) + \sum_{c \in C_l} (u+3r)(c) + \sum_{c \in C_2} (u+3s)(c)$ with $C_l \oplus C_2 = [l,n] \cup \{b\}$

$. (\mu_{2,1} + 3\mu_{3,s(a)})$

$-(u+3s)(a) + \sum_{c \in C_l} (u+3r)(c) + \sum_{c \in C_2} (u+3s)(c)$ with $C_l \oplus C_2 = [l,n] \cup \{b\}$

$. (\mu_{2,1} + 3\mu_{0,2a})$

$-(u+3s)(a) + \sum_{c \in [l,n] \cup \{a,b\}} u(c)$

If we keep only the minimal solutions we obtain the generative family:

$.(p+r)(i)$ $\forall\, i \in [l,n] \cup \{a,b\}$

$.(p+s)(i)$ $\forall\, i \in [l,n] \cup \{a,b\}$ $(i \neq a)$

$.(p+s)(a) + \sum_{c \in [l,n] \cup \{a,b\}} p(c)$

$.(u+3s)(a) + \sum_{c \in [l,n] \cup \{a,b\}} u(c)$

$\sum_{c \in C_l \cup \{a\}} (u+3r)(c) + \sum_{c \in C_2} (u+3s)(c)$ with $C_l \oplus C_2 = [l,n] \cup \{b\}$

IV-2 Unary regular nets

The computation of positive semi-flows in unary regular net is similar to the one in unary predicate/transition net, but much more technical because we work now on polynomials. Also we only give the principle of the algorithm; for more details, please refer to [CHP90].

As we see in section III-2.1, the computation of positive flows in unary regular net is equivalent to solve the system (2) :

$$(2)\quad \begin{cases} (D+nB).\sum_{j=l}^{n} X_j=0 \\ D.X_1=D.X_2=...=D.X_n \end{cases}$$

Let us suppose that b_j is a vector such that $Sol(A,b_j) = \{V_{j,k}\}$. Each vector of $Sol^n(A,b_j)$ is a solution of the second equation $D.X_l=...=D.X_n$. These solutions can be expressed as a formal sum:

$$X=\sum_{c\in C_{j,k}} V_{j,k}(c) \text{ with } \oplus C_{j,k}=C \text{ if } b_j\neq 0 \text{ and } X=V_{0,k} \text{ if } b_j=0$$

In order to solve the system (2) we have to report the solutions of the equation $D.X_l = \ldots =D.X_n$ to the first one $(D+nB).\sum X_j=0$. So for each b_j, if we denote by $k_{j,k}$ the size of $C_{j,k}$, $k_{j,k} = |C_{j,k}|$, we obtain the formal vector:

$$W_j = \sum k_{j,k}(D+nB).V_{j,k} \text{ with } \sum k_{j,k}= n$$

And for $b_j=0$ we have for all $V_{0,k}$ in $Sol(D,0)$ the formal vector:

$$W_0^{\,k} = (D + nB).V_{0,k}$$

Also we have to solve the system with $S = \{b_j\}$ the set corresponding to the matrix D computed by the algorithm 3 and for each b_j in S, $Sol(D,b_j) = \{V_{j,l},...,V_{j,mj}\}$:

$$\sum_{V_{0,k}\in Sol(D,0)} \mu_k^{\,0} . W_k^{\,0}+ \sum_{bj\in S\backslash\{0\}}\mu_j.W_j = 0 \text{ where the unknows are } \mu_k^{\,0}, \mu_j$$

which corresponds to:

$$\sum_{V_{0,k}\in Sol(D,0)} \mu_k^{\,0} . (D+n.B).V_{0,k}+ \sum_{bj\in S\backslash\{0\}}\mu_j. \sum_{k=l,..,mi} k_{j,k}(D+n.B).V_{j,k} = 0 \text{ with for each } j\neq 0 \sum_{k=l,..,mi} k_{j,k}=n$$

As for each bj and each vector $V_{j,k}$ of $Sol(D,b_j)$, $D.V_{j,k} = b_j$, it is equivalent to solve the system:

$$\sum_{V_{0,k}\in Sol(D,0)} \mu_k^{\,0} . (0+n.B.V_{0,k})+ \sum_{bj\in S\backslash\{0\}}\mu_j. (n.b_j + \sum_{k=l,..,mi} n.k_{j,k}.B.V_{j,k}) = 0 \text{ with for each } j\neq 0 \sum_{k=l,..,mi} k_{j,k}=n$$

which is also equivalent to:

$$\sum_{V_{0,k}\in Sol(D,0)} \mu_k^{\,0} . (B.V_{0,k})+ \sum_{bj\in S\backslash\{0\}}\mu_j. (b_j + \sum_{k=l,..,mi} k_{j,k}.B.V_{j,k}) = 0 \text{ with for each } j\neq 0 \sum_{k=l,..,mi} k_{j,k}=n$$

This last system can be solved by the Farkas' algorithm extended to generic families [CHP90]. Let us see on an example how this algorithm works.

We consider the example given in part II-I (figure 1) and we prove that the place *Repos* is an implicit place. Also we consider the net in which we have reversed the arcs adjoining to the place Repos, and we compute then a generative family of semi-flows on this net.

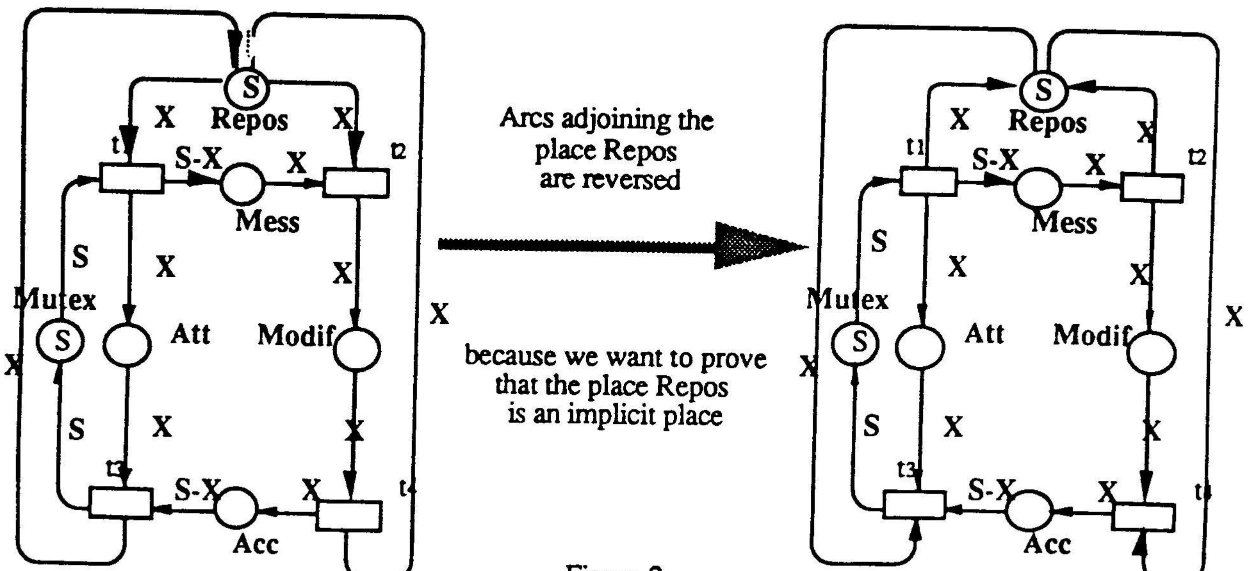

S
X
Repos
X
t1
S-X
X
t2
Mess
S
X
X
Mutex
S
Att
Modif
X
X
S
X
X
t3
S-X
X
t4
Acc
Arcs adjoining the place Repos are reversed
because we want to prove that the place Repos is an implicit place

Figure 2

We have the matrix D and B as defined in part III-2.1:

$$\begin{array}{c|cccccc|c} & \text{Repos} & \text{Mess} & \text{Modif} & \text{Mutex} & \text{Att} & \text{Acc} & \\ \hline & 1 & -1 & 0 & 0 & 1 & 0 & t_1 \\ D = & 1 & -1 & 1 & 0 & 0 & 0 & t_2 \\ & -1 & 0 & 0 & 0 & -1 & 1 & t_3 \\ & -1 & 0 & -1 & 0 & 0 & 1 & t_4 \end{array}$$

$$\begin{array}{c|cccccc|c} & 0 & 1 & 0 & -1 & 0 & 0 & t_1 \\ B = & 0 & 0 & 0 & 0 & 0 & 0 & t_2 \\ & 0 & 0 & 0 & 1 & 0 & -1 & t_3 \\ & 0 & 0 & 0 & 0 & 0 & 0 & t_4 \end{array}$$

So we solve and the first equation $D.X_1=...=D.X_n$ and we obtain the family S defined by (following the order t_1,t_2,t_3,t_4):

S = ((0), b_1=(0,1,0,-1), b_2=(0,0,1,1), b_3=(1,0,-1,0), b_4=(-1,-1,0,0), b_5=(1,1,-1,-1) } with

Sol(A,(0)) = { (Mutex), (Mess+Modif+Att+Acc), (Repos+Mess+Acc) }
Sol(A,b_1) = {(Modif)}
Sol(A,b_2) = { (Acc) }
Sol(A,b_3) = { (Att) }
Sol(A,b_4) = { (Mess) }
Sol(A,b_5) = { (Repos), (Modif+Att) }

For each b in S we have the projection on the equation $(D+nB)\Sigma X$:

(0) gives 3 vectors:

.$f_{0,1} =(-1,0, 1,0)$, $f_{0,2} =(1,0, -1, 0)$, $f_{0,3} =(1, 0, -1, 0)$

b_1 gives the vector:

.$f_1 =(0, 1,0, -1)$

b_2 gives the vector:

.$f_2 =(0, 0, 1-n, 1)$

b_3 gives the vector :

.$f_3 =(1,0, -1, 0)$

b_4 gives the vector:

.$f_4 =(n-1, -1, 0, 0)$

b_5 gives the vector:

$(1, 1, -1, -1) + k_1(0,0,0,0) + k_2(0,0,0,0)$ with $k_1+k_2=n$, so:

.$f_5 =(1, 1, -1, -1)$

Also we have to solve the system W.X= 0 with W the matrix:

$$W = \begin{array}{cccccccc} f_{0,1} & f_{0,3} & f_{0,2} & f_1 & f_2 & f_3 & f_4 & f_5 \\ \hline -1 & 1 & 1 & 0 & 0 & 1 & 1 & n-1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 \\ 1 & -1 & -1 & 0 & 1-n & -1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \end{array}$$

We "nullify" the second and the fourth row of the preceeding matrix and we obtain:

$$W = \begin{array}{cccccc} f_{0,1} & f_{0,3} & f_{0,2} & f_3 & f_1+f_2+f_5 & f_2+f_4+f_5 \\ \hline -1 & 1 & 1 & 1 & n-1 & n \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & -1 & 1-n & n \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

As the polynomial (n-l) and n have a constant sign, $n\geq 2$, if we "nullify" the first row we obtain the following matrix:

$$W = \begin{array}{ccccc} f_{0,1}+f_{0,2} & f_{0,1}+f_{0,3} & f_{0,1}+f_3 & (n-1)f_{0,1}+f_1+f_2+f_5 & nf_{0,1}+f_2+f_4+f_5 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}$$

At this step we have obtained a generative family which gives the following semi-flows:

.(Mutex+Repos+Mess+Acc)(i) for any i in [1,n]

.(Mutex+Mess+Modif+Att+Acc)(i) for any i in [1,n]

.(Mutex)(i) $+\sum_{c \in C}$(Att)(c) for any i in [1,n]

.(n-1).(Mutex)(i) $+\sum_{c \in C}$ (Modif+Acc)(c)+ $\sum_{c_1 \in C_1}$ (Repos)(c_l) + $\sum_{c_2 \in C_2}$(Modif+Att)(c_2)

for any i in [1,n] and where $\{C_1, C_2\}$ is a partition of C

.n.(Mutex)(i) + $\sum_{c \in C}$(Mess+Acc)(c)+ $\sum_{c_1 \in C_1}$(Repos)(cl) + $\sum_{c_2 \in C_2}$(Modif+Att)(c2)

for any i in [1,n] and where $\{C_1, C_2\}$ is a partition of C

This last flow gives the particular flow(if we take $C_2 = \emptyset$):

.n.(Mutex)(i) + $\sum_{c \in C}$(Mess+Acc+Repos)(c) for any i in [1,n]

This flow ensures that Repos is an implicit place [Had87,p.166].

Conclusion

We have proposed two algorithms for a computation of a generative family of positive semi-flows in two basic families of coloured nets. These two algorithms are based on the resolution of the parametrized equation $A.X_l = ... = A.X_n$, where n is the parameter of the model, A a matrix and X_i the unknowns are vectors. We have also proposed an algorithm which solves this equation. The general idea of this algorithm is that the resolution of the system for a sufficiently great n provides the general form of a generative family for any n.

The perspective of this work is of course the computation of positive semi-flows in more complex systems. In this way, there are two possibilities: on the one hand, similar systems but with many parameters -like regular nets or predicate/transition nets -and on the other hand, systems with more complex structure but with only one parameter -like ordered nets -Another perspective of this work can also be the computation of others linear invariants such like deadlocks and traps.

BIBLIOGRAPHY

[Aze85] : P.Azema, P.Ladet, J.Martinez, M.Silva
"Modelling and validation of complex systems by coloured Petri nets". Fifth European whorkshop on applications and theory of Petri nets. Aarthus, Denmark (1984)

[Bra83] : G.W. BRAMS
"Réseaux de Petri. Théorie et pratique". Masson editeur, Paris, (1983)

[CHP90] : J.M. Couvreur, S. Haddad, J.F. Peyre
"Resolution paramétrée d'une famille de systèmes d'équations linéaires a solutions positives" Rapport I.B.P., Université P.& M. Curie et C.N.R.S. MAS I, 4 place Jussieu, Paris (1990)

[Col89] : J.M. Colomb, M.Silva
"Improving the linearly based Characterization of P/T nets". 10th International Conference on Application and Theory of Petri Nets. Bag Godesberg, June 89 (1989)

[Cou90] : J.M. Couvreur
"The general computation of flows for coloured nets". 11th International Conference on Application and Theory of Petri Nets. Paris, June 90 (1990)

[Far02] : J.Farkas
"Theorie der einfahren Ungleichungen". Journal fliT reine und angew. Mathematik 124, pp 1-27 (1902)

[Gen79] : H.J. Genrich, K.Lautenbach
"The analysis of distributed systems by means of Predicate/Transition nets". Semantics of concurrent computation. Lecture notes in Computer science N°70 Springer-Verlag (1979)

[Gen81] : H.J. Genrich, K. Lautenbach
"System modelling with high-level Petri nets". Theoretical computer science 13, 1981, pp 103. 136 (1981)

[Gen82] : H.J. Genrich, K. Lautenbach
"S-Invariance in predicate transistion nets" .Third european workshop on applications and theory of Petri nets. Varennes Italy (1982)

[Had86] : S. Haddad, C. Girault
"Algebraic structure of flows of a regular net". Seventh european whorkshop on applications and theory of Petri nets, Oxford England, June 1986, in "Advances in Petri nets 87", L.N.C.S 266, G.Rosenberg (Ed.), Springer Verlag,1897, pp 73-88 (1987)

[Had87] : S .Haddad
"Une catégorie réguliere de réseau de Petri de haut niveau: définition, propriétés et réductions Application à la validation de systèmes distribués". Thèse de l'Université Pierre et Marie Curie Paris, Juin 87 (1987)

[Hub84]: P.Huber, A.M. Jensen, L.O. Jepsen, K. Jensen
"Towards reachability trees for high-level Petri nets"Fifth european whorkshop on applications and theory of Petri nets. Aarthus, Denmark (1984)

[Jen81] : K.Jensen
"Coloured Petri nets and the invariant method". T.C.S. 14, pp 317-336 (1981)

[Jen86] : K. Jensen
"High-level Petri nets. A combination of Predicate/Transition nest and coloured nets" Advanced course on Petri nets. Bad Honnef (1986)

[Lau85] : K.Lautenbach, A.Pagoni
"In variance and duality predicate transition nets and in coloured nets". Arbeitspapiere der G.M.D. 132 (1985)

[Mem80] : G.Memmi and G.Roucairol

"Linear algebra in net theory". Proc. of "Advanced course on general Net Theory of Processes and Systems" Hambourg 1979, L.N.C.S. 84, W.Brauer (Ed.), Springer Verlag (1980)

[Mem83] : G.Memmi
"Méthode d'analyse de réseaux de Petri, réseaux à files, et applications aux systèmes temps réel". These de Doctorat d'Etat, Université Pierre et Marie Curie, Paris, Juin 83 (1983)

[Pey90] : J-F. Peyre
"Recherche d'invariants lineaires dans les systèmes parametres". Magistere MMFAI, E.N.S. Ulm et Universités Paris 6, Paris 7, Paris 11, Paris, Novembre 90 (1990).

[Siv85] : M.Silva, J.Martinez, P.Ladet, H.Alla
"Generalized inverses and the calculation of invariants for coloured Petri nets". Technique et science informatique Vol.4 N°l (1985)

[Tre86] : N. Treves
"Le calcul d'invariants dans les reseaux a Prédicats/Transitions unaires". These de l'université Paris Sud , Paris (1986)

[Vau85] : J. Vautherin and G.Memmi
"Computation of flows for unary Predicates/Transitions nets". in "Advances in Petri Nets 1984", L.N.C.S. 188, G.Rosenberg (Ed.), Springer Verlag, pp 307-327 (1985)

[Vau86] : J. Vautherin
"Calculation of semi-flows of Pr/T-systems". Research Repon L.R.I, N° 130, Universite Paris-Sud, Octobre 1986 (1986)