An Algorithm to Compute a Basis of Petri Net Invariants

S. Cayir and M. Ucer
Electronics and Communication Department, Istanbul Technical University, Istanbul, Turkey
cayirs@itu.edu.tr and murvet@ehb.itu.edu.tr

Abstract - A fast, simple and effective method to compute a basis of Petri Net invariants is presented in this paper. Another algorithm to obtain a basis with only positive elements is also initiated. Both of the algorithms are based on simple row operations. Where as the second algorithm is a modified form of the first one under specific constraints. Results of the algorithms applied on an example net are covered and the running times for different sized nets are given.

INTRODUCTION

Petri Nets are known as one of the best defined approaches to modeling of discrete and concurrent systems. Petri Nets are a graphical and mathematical modeling tool used to model and analyze information processing systems, communication systems and protocols, real-time systems, multi-processor systems, automatic production systems, flow charts, chemical reactions, ecological systems which may show concurrent, asynchronous, distributed, parallel, non-deterministic and/or stochastic behavior [1], [2].

Using Petri Nets in system design takes the steps of modeling, analysis and implementation. Modeling step includes graphical visualization of complex systems, while implementation step is comprised of well defined synthesis procedures. Petri Net analysis step includes system analysis of structural and behavioral properties. Reachability, boundedness, liveness, reversibility, coverability, persistence and fairness are the most known behavioral properties. Behavioral properties rely on the initial state of the Petri Net, whereas structural properties depend on the topology of the net. Structural boundedness, conservativeness, controllability, structural liveness, and consistency are the most known structural properties. Murata in [2] argues structural and behavioral properties of Petri Nets in detail.

Properties such as boundedness and conservativeness can be completely determined by means of Petri Net invariants. While reachability and liveness properties can be tested using Petri Net invariants. A detailed discussion on the relation between controllability and boundedness of Petri Nets based on invariants can be found in [3]. Many definitions of place invariants and transition invariants have been made in the literature [4-10].

In this paper, the main concern is to find a basis of invariants which figure a powerful tool to study structural properties of Petri Nets. Several methods have been proposed in the literature. These methods can be classified in two groups; algorithms to obtain all and/or minimal positive invariants in [4] and [5], and algorithms to obtain basis of invariants in [6] and [7]. The method proposed in this study focuses on finding a basis of invariants based on simple row operations applied on the incidence matrix. The first method is focused to find basis without consideration if the results are positive or negative. Then a more restricted approach to find a basis with positive or semi-positive elements is researched. The constraints and necessary conditions of the approach are also established. Though the method is simple it is also effective, fast and open to further development.

NOTATION AND DEFINITIONS

A generalized Petri net is a 5-tuple, PN = (P, T, F, W, M_0) where,

$P = \{p_1, p_2 \dots p_m\}$ is a finite set of places,

$T = \{t_1, t_2 \dots t_n\}$ is a finite set transitions,

$F \subseteq (PxT) \cup (TxP)$ is set of directed arcs defining the relations between the places and the transitions,

$W: F \rightarrow \{1, 2, 3 \dots\}$ is a weight function,

$M: P \rightarrow \{0, 1, 2 \dots\}$ is the initial marking,

$P \cup T \neq \varnothing$ and $P \cap T = \varnothing$, m is the number of places and n is the number of transitions. [2]

The graphical representation of a place p_i is with a circle, whereas a transition t_j is represented with a box or bar. The relation between a place and a transition is represented with a directed weighted arc. The weight of an arc from place p_i to transition t_j is represented as $w(p_i, t_j)$, whereas an arc in reverse direction is represented with $w(t_j, p_i)$.

The dynamic behavior of a system is modeled through flow of tokens on the net. A marking M assigns a nonnegative integer to every place. M is m-vector,

where m is the number of places. $M(p_i)$ is the number of tokens in place p_i. M_0 is the initial marking. A marking M_k corresponds to the state of the net at a given moment. A transition t_j is said to be enabled at marking M in case all of the input places have at least as many tokens as weight of the arc from the input places to that transition. After a transition is fired under marking M_k, distribution of the tokens changes according to weights of the arcs. And the state changes to M_{k+1}. [2]

The change of states in Petri Nets is captured with State Equation. The state equation is based on algebraic equation 1.

$$\mathbf{M} = \mathbf{M_0} + \mathbf{A}.\mathbf{u} \quad (1)$$

Where, $\mathbf{M_0}$ is the initial marking.

M is the marking reachable from $\mathbf{M_0}$ after consecutive transition firings.

A is the incidence matrix. It is nxm matrix of integers. The a_{ij}'th element of the incidence matrix is computed in 2.

$$a_{ij} = w(t_i, p_j) - w(p_j, t_i) \quad (2)$$

i = 1, 2, …, n, j = 1, 2, …, m, n and m denotes the number of transitions and places respectively.

u is called the *firing count* vector. It is nx1 vector of non-negative integers. The ith element of σ implies how many times t_i transition must be fired in a sequence of firings.

P-invariants are integer solutions of homogenous equation 3.

$$\mathbf{A}.\mathbf{x} = \mathbf{0} \quad (3)$$

Place invariants formalize invariant properties regarding places in Petri Net. For example in a set of places the sum of tokens remains unchanged after firing. Then this set can define a place invariant. A good discussion of various implications of P-invariants can be found in [8] and [9].

T-invariants are positive integer solutions of homogenous equation 4.

$$\mathbf{A^T}.\mathbf{x} = \mathbf{0} \quad (4)$$

T-invariants are especially useful when the solution of 4 is positive, in such case x corresponds to a cyclic sequence. By that means T-invariants define invariant properties regarding firing sequences applicable to a Petri Net. They are useful to define consistent components with in a net. When a net is not consistent it can not be lively and boundedly marked [10].

Computation of invariants is based the solutions of homogenous equations 3 and 4. Actually equation 4 is the dual of equation 3. Here after only the solution of P-invariants will be considered. T-invariants can be calculated by applying the same computing steps on the inverse of the incidence matrix **A**.

PROPOSED ALGORITHMS

First, an algorithm to compute a basis of invariants in generalized Petri nets that derives invariants with both positive and negative elements is considered. This algorithm is based on theorem 4 covered in the previous section. Then verbal definition of the second algorithm used to obtain positive and/or semi positive only is given. This algorithm is applicable only if the condition defined in theorem 5 holds.

Algorithm 1: Computation of a basis for P - invariants

Input: The mxn dimensional incidence matrix **A**, m = |P|, n = |T|

Output: The matrix **B** whose rows form the basis

Step 1: Take the transpose of the incidence matrix **A**. Define nxn dimensional identity matrix $\mathbf{I_n}$.

Step 2: for i = 1 to m;

1. Apply simple row operations on $\mathbf{A^T}$. These row operations shall form the $\mathbf{A^T}$ matrix an upper triangular form.

2. Apply the same row operations on the Identity matrix $\mathbf{I_n}$.

Step 3: Abstract the last (n-r) rows as matrix **B** from identity matrix $\mathbf{I_n}$ which correspond to zero rows on $\mathbf{A^T}$. The rows of **B** form a basis for P – invariants.

Flow chart of algorithm 1 can be seen in Fig. 1.

Algorithm 2: Computation of a positive basis for P - invariants

Input: The mxn dimensional incidence matrix **A**, m=|P|, n=|T|

Output: The matrix B whose rows form the basis

Step 1: Take the transpose of the incidence matrix **A**. Define nxn dimensional identity matrix $\mathbf{I_n}$.

Step 2: for i = 1 to m;

1. Apply additive simple row operations on $\mathbf{A^T}$. These row operations shall form the $\mathbf{A^T}$ matrix an upper triangular form.

2. Apply the same additive row operations on the identity matrix $\mathbf{I_n}$.

Step 3: Abstract the last (n-r) rows as matrix **B** from identity matrix $\mathbf{I_n}$ which correspond to zero rows on $\mathbf{A^T}$. The rows of **B** form a positive basis for P – invariants.

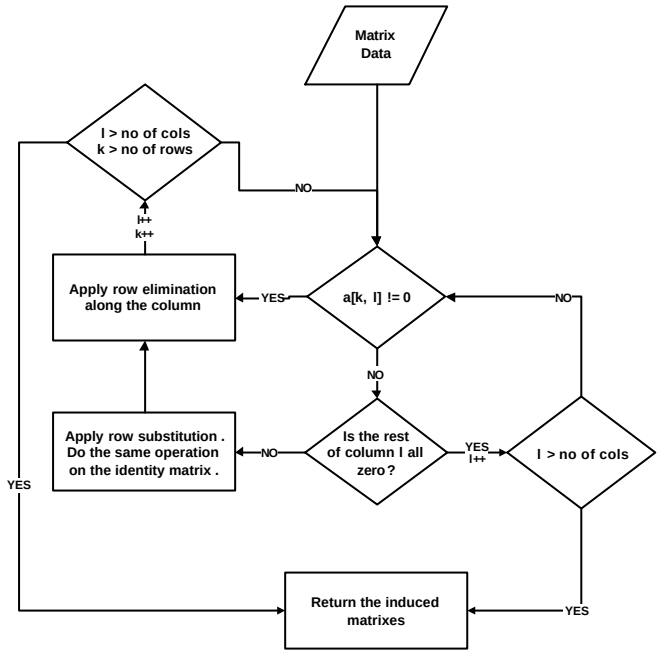

Figure 1: Algorithm 1 Flow Chart

In order to obtain a basis for T – invariants the same steps will be applied on the incidence matrix **A**. This holds for both algorithms.

The code developed with regard to the algorithms defined has the following pseudo code structures.

Pseudo Code for Algorithm 1

Input: An incidence matrix **a**, nxm dimensional
Output: (n-r)xn dimensional basis matrix **b**

begin

1. **define** id /* nxn dimensional identity matrix */
2. **while** (k < n) **and** (l < m)
3. **if** (a(k,l) != 0)
4. **for** (i = k+1; i < n)
5. RowElimination(a, id);
6. k++, l++;
7. **else if** (RestofColumnZero(a) == false)
8. ExchangeWithNonZeroRow(a, id);
   /* exchange the row with a non zero one */
9. for(i = k+1; i < n)
10. RowElimination(a, id);
11. k++, l++;
12. **else if** (RestofColumnZero(a) == true)
    /* checks if the rest of he column is zero */
13. l++

end

end

The pseudo code for subroutine Row Elimination is as follows,

Subroutine RowElimination(a, id)

begin

1. **if** (a(k,l).a(i,l) < 0)
2. a(j,i) ← a(j,i).a(k,l) + a(k,i).a(j, l)
3. id(j,i) ← id(j,i).a(k,l) + id(k,i).a(j, l)
4. **else if** (a(k, l).a(i,l) > 0)
5. a(j,i) ← a(j,i).a(k,l) - a(k,i).a(j, l)
6. id(j,i) ← id(j,i).a(k,l) - id(k,i).a(j, l)

end if

end

The pseudo code for Algorithm 2 is the same except at steps 5 and 10, where positive row elimination subroutine is applied. The subroutine of positive row elimination is as follows;

Subroutine PositiveRowElimination (a, id)

begin

1. **if** (a(k,l) > 0)
2. FindFirstNegative (a(k))
   /* finds first negative row */
3. EAPR_FirstNegative (a, id)
   /* eliminates all positive rows with the first negative one*/
4. EANR_FirstRow (a, id)
   /* eliminates all negative rows with the first row */
5. **else if** (a(k, l) < 0)
6. FindFirstPositive (a(k))
   /* find first positive row */
7. EANR_FirstPositive(a, id)
   /* eliminate all negative rows with first positive row */
8. EAPR_FirstRow(a, id)
   /* eliminate all positive rows with the first row */

end if

end

AN EXAMPLE

The algorithms are applied to the Petri Net in Fig. 2. The incidence matrix of the net follows.

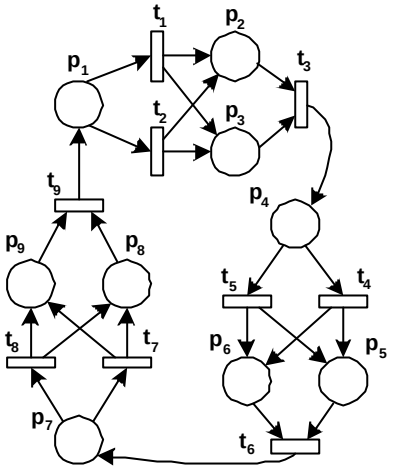

Figure 2: An example Petri Net

$$\begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

The proposed invariant finding algorithms are applied. The example Petri Net has 9 places and the rank of the incidence matrix is 5, so the resultant basis is comprised of four row vectors. Results of the standard analysis has both positive and negative elements, where as results of the positive analysis has only positive elements.

$$\begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Positive only P – invariant basis;

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Notice out that there can be found at least one invariant covered by positive elements.

EXPERIMENTAL RESULTS

A computer program implementing both standard and positive algorithms was developed. The computer program was named Petri Net Invariant Analysis (PNIA). It was developed with C#. PNIA generates basis' both for P – invariants and T – invariants. Invariants of different sized Petri Nets were executed with PNIA on a computer with Pentium 4 2.4GHz processor and 512MB memory. Results are listed in Table 1.

Size of A (mxn)	P analysis	Positive P analysis	T analysis	Positive T analysis
9x9	4,56E-05	6,58E-05	4,04E-05	6,03E-05
18x18	1,39E-04	3,74E-04	1,48E-04	3,83E-04
36x36	5,38E-04	2,59E-03	5,82E-04	2,63E-03
72x72	2,29E-03	1,84E-02	2,48E-03	1,96E-02
144x144	9,70E-03	1,39E-01	1,06E-02	1,49E-01
216x216	2,20E-02	4,15E-01	2,28E-02	4,45E-01

Table 1: Running time results (in s)

CONCLUSION

In this paper, a new approach to determine invariant basis of Petri Nets have been presented. Using the bases' the complete set of P – invariants and T – invariants can be generated. Both of the proposed algorithms have computing time complexities of $\Theta((M+N).N)$. The positive invariant analysis is slower because of the more controls present in its algorithm. When the results from the previous section are examined the computing time complexity shows the pattern of $\Theta((M+N).\log N)$.

Further improvements can be made on the algorithm to find solutions with more constraints. Here the constraint was to find positive basis. Other constraints can be to find a semi positive basis or a minimal basis. The resultant sets of invariants can be used to find all of the minimal invariants or any other type of invariant regarding the goal of the applications.

REFERENCES

[1] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Inc, Englewood Cliffs, 1981

[2] T. Murata, Petri Nets: Properties, Analysis and Applications, *Proceeding of the IEEE*, 1989, Vol. 77, No.4, 541-580

[3] P. Ramachandran, M. Kamath, On Place Invariant Sets and the Rank of the Incidence Matrix of Petri Nets, *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, San Diego, USA, 1998, pp. 160-165

[4] J. Martinez, M. Silva, A simple and fast algorithm to obtain all invariants of a generalized Petri Net, 2nd Eu. Ws. on app. and theo. of PN, Bod-honneff, 1981

[5] K. Takano, S. Taoka, M. Yamauchi, T. Watanabe, T., Two Efficient Methods For Computing Petri Net Invariants, *IEEE Int. Conf. on Sys., Man, and Cybernetics*, Tucson, USA, 2001, Vol. 4, 2717–2722

[6] A. Bourjij, M. Boutayeb, D. Koenig, T. Cecchin, On Generating a Basis of Invariants in Petri Nets, *IEEE Int. Conf. on Sys., Man, and Cybernetics, Comp. Cyber. and Simulation*, Orlando, USA, 1997, Vol.3, 2228-2233

[7] T. Tanida, T. Watanabe, K. Onega, A Polynomial-Time Algorithm For Finding a Semi-Gen. of Petri Net Invariants, *IEEE Int. Sym. on Cir. and Sys.*, Singapore, 1991, Vol. 5, 2838-2841

[8] J. Desel, W. Reising, Place or Transition Petri Nets, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, Springer, 1998, pp. 122-173

[9] J. Desel, Basic Linear Algebraic Techniques for Place/Transition Nets, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, Springer, 1998, pp. 257-308

[10] M. Silva, E. Teruel, J. M. Colom, Linear Algebraic and Linear Programming Tech. for the Anal. of Place/Trans. Net Sys., *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, Springer, 1998, pp.309-373