Petri nets analysis using incidence matrix method inside ATOM^3

Alejandro Bellogín Kouki
Universidad Autónoma de Madrid
alejandro . bellogin @ uam . es

June 13, 2008

Outline

Introduction

Analysis methods for Petri nets may be classified into the following three groups:

1. Coverability (reachability) tree method
2. Matrix-equation approach
3. Reduction or decomposition techniques

Incidence matrix

Given a Petri net (from now on PN), if we want to analyse it from an algebraic point of view, we need to get its *incidence matrix*.

Definition

It is an $n \times m$ matrix, for a PN with n transitions and m places, and its typical entry is given by

$$a_{ij} = a_{ij}^{+} + a_{ij}^{-}$$

where $a_{ij}^{+} = w(i, j)$ is the weight of the arc from transition i to its output place j, and $a_{ij}^{-} = w(j, i)$ is the weight of the arc from transition i from its input place j.

Incidence matrix: state equation

Definition

$$M^k = M^{k-1} + A^t u_k$$

where $A = [a_{ij}]$ is the incidence matrix, u_k is an $n \times 1$ column vector of $n - 1$ 0's and one nonzero entry, indicating the transition fired, and M^j is the net marking after j transitions have been fired.

Incidence matrix: applications

We are going to explain how we have used these matrices in order to obtain results with using the techniques herewith listed:

- Subclassification or detection of particular structures
- Invariant analysis
- Detection of siphons and traps
- Reachability to a marking

Incidence matrix: applications

We are going to explain how we have used these matrices in order to obtain results with using the techniques herewith listed:

- Subclassification or detection of particular structures
- Invariant analysis
- Detection of siphons and traps
- Reachability to a marking

Structural properties

- Subclassification
- Detection of siphons and traps

Tools

Name	Functionality	Finally used
Matlab	System solver	No
Numarray	Python's library to work with matrices	Yes
Pipe	Tool for creating and analysing PNs	Yes
slepc4py	Python's library to wrap SLEPc eigensolver package	No

Table: Tools or libraries checked

Outline

Subclassification of Petri nets

We use the following notation:

$\bullet t$	$=$	$\{p : (p,t) \in F\}$	is the set of input places of t
$t\bullet$	$=$	$\{p : (t,p) \in F\}$	is the set of output places of t
$\bullet p$	$=$	$\{t : (t,p) \in F\}$	is the set of input transitions of p
$p\bullet$	$=$	$\{t : (p,t) \in F\}$	is the set of output places of p

All Petri nets considered in this section are ordinary, what means that all of its arc weights are 1's.

State machine

In this kind of net, each transition has exactly one input place and exactly one output place:

$$|\bullet t| = |t\bullet| = 1 \forall t \in T$$

State machine

In this kind of net, each transition has exactly one input place and exactly one output place:

$$|\bullet t| = |t \bullet| = 1 \forall t \in T$$

This condition, given the input and output matrices, is very simple:

$$\forall i \quad \sum_j |a_{ij}^{\triangleright}| = 1, \quad \triangleright \in \{-, +\}$$

In other words, in all the *rows* of input and output matrices must be a nonzero element, and only one.

Marked graph

Each place has exactly one input transition and exactly one output transition:

$$|\bullet p| = |p \bullet| = 1 \forall p \in P$$

Marked graph

Each place has exactly one input transition and exactly one output transition:

$$|\bullet p| = |p\bullet| = 1 \forall p \in P$$

This is very similar to the previous case:

$$\forall j \quad \sum_i |a_{ij}^{\triangleright}| = 1, \quad \triangleright \in \{-,+\}$$

Equivalent to the following restriction: in all the *columns* of input and output matrices must be a nonzero element, and only one. Actually, this property is equivalent to check if the *transpose net* is a state machine.

Free-choice net

In these nets, every arc from a place is either a unique outgoing arc or a unique incoming arc to a transition:

$$\forall p \in P, |p \bullet| \leq 1 \text{ or } \bullet(p\bullet) = \{p\}$$

Free-choice net

In these nets, every arc from a place is either a unique outgoing arc or a unique incoming arc to a transition:

$$\forall p \in P, |p \bullet| \leq 1 \text{ or } \bullet(p\bullet) = \{p\}$$

We can not express this condition in a simpler way, but we can give an algorithm:

- For each place in the input matrix (output place is equivalent to input transition):
  - If it has more than one element nonzero (the place has two or more output transitions):
    - For each transition from this place, check that it has only one element nonzero in its row (its only input is this place). If it is not true, the net is not a free-choice net.

Extended free-choice net

It is a PN such that

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow p_1 \bullet = p_2 \bullet \quad \forall p_1, p_2 \in P$$

Extended free-choice net

It is a PN such that

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow p_1 \bullet = p_2 \bullet \quad \forall p_1, p_2 \in P$$

Like in the previous case, we can only give an algorithm:

- For each place p in the input matrix:
    - For each place $q \neq p$:
        - Check that if their set of output transitions have empty intersection. If it is the case, continue. If not, the intersection has to be equal to the set of output transition of each place, i.e. both places must have the same output transitions, if it is not true, this net is not an extended free-choice net.

Asymmetric choice net

This type of net is characterised by the following equation:

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow p_1 \bullet \subseteq p_2 \bullet \text{ or } p_1 \bullet \supseteq p_2 \bullet \quad \forall p_1, p_2 \in P$$

Asymmetric choice net

This type of net is characterised by the following equation:

$$p_1 \bullet \cap p_2 \bullet \neq \emptyset \Rightarrow p_1 \bullet \subseteq p_2 \bullet \text{ or } p_1 \bullet \supseteq p_2 \bullet \quad \forall p_1, p_2 \in P$$

This is a more relaxed form of the previous net, where the last *equal* condition is smoothed by the subset condition:

- For each place p in the input matrix:
    - For each place $q \neq p$:
        - Check that if their set of output transitions have empty intersection. If it is the case, continue. If not, the set of output transition of one place must contain the other, or viceversa, if it is not true, this net is not an asymmetric choice net.

Invariant analysis I

Definitions

- x is a T-invariant (transition invariant) if

$$A^t x = 0$$

- y is a P-invariant (place invariant) if

$$Ay = 0$$

Invariant analysis II

At first sight, we thought this problem was a simple eigenvector problem, where the invariants are the eigenvectors. But:

Problems

1. The incidence matrix is not, generally, a square matrix.
2. Even in the square matrix situation, the eigenvector found does not have the information we want.

[1] pipe2.sourceforge.net

Invariant analysis II

At first sight, we thought this problem was a simple eigenvector problem, where the invariants are the eigenvectors. But:

Problems

1. The incidence matrix is not, generally, a square matrix.
2. Even in the square matrix situation, the eigenvector found does not have the information we want.

We use PIPE library[1] to find the minimal generating set of all place invariants, since it worked properly in all tested examples. It uses the algorithm proposed by D'Anna and Trigila in [1] (very good performance).

[1] pipe2.sourceforge.net

Siphons and traps I

Definitions

- **Siphon** (or *deadlock*) is a set of places S such that every transition having an output place in S has an input place in S:

$$\bullet S \subseteq S \bullet$$

- **Trap** is a set of places Q where every transition having an input place in Q has an output place in Q:

$$Q \bullet \subseteq \bullet Q$$

The word *set* implies the calculations over all the possible groups of places that can be created. Fortunately, the order is not important.

Siphons and traps II

The total number of sets that have to be checked is

$$\sum_{k=1}^{m} \binom{m}{k} = \sum_{k=1}^{m} \frac{m!}{k!(m-k)!} = 2^m - 1, \text{ with } m = |P|$$

For each set of places, the sets of input and output transitions are generated. At this point, a simple comparison between sets (operation *subset of*) is carried out, and depending on the answer, we have a trap, a siphon or nothing at all

Example

In the problem of philosophers, our program finds 255 siphons and 255 traps when 4 philosophers come into play (this net has 8 transitions and 10 places).

Reachability to a marking I

Motivation

The problem of generating the reachability (coverability) graph of a specific PN is a very complex and time consuming task.

Reachability to a marking II

Algorithm

- Find all the sequences $\vec{U}$ that can potentially reach to the marking M
- Take into account the order

Reachability to a marking II

Algorithm

- Find all the sequences $\vec{U}$ that can potentially reach to the marking M
- Take into account the order

Combinatorial problems

- Calculate all the possible firing sequences. At this point we have to limit the number of times each transition can be executed.
- Calculate all the permutations of a given sequence of transitions, to consider the order.

Reachability to a marking: performance

$\|P\|$	$\|T\|$	L	Time 1 (secs.)	Time 2 (secs.)	Valid sequences found	Max. length of sequence
3	4	1	1	1	1	3
3	4	3	1	4	1	7
3	4	5	1	38	2	8
3	4	10	1	38	2	8
4	3	1	1	1	1	2
4	3	3	1	1	1	6
4	3	5	1	540	1	9
4	3	10	2	460	0 (HE)	0
10	8	1	1	2 (HE)	8	7
10	8	2	1	860 (HE)	17	9
10	8	3	1	754 (HE)	11	7

Table: Time spent to obtain a solution for the *reachability to a marking* problem (if any). *Time 1* is the time to get all possible firing sequences, while *Time 2* is the time spend to check if the ordered sequences are valid. HE means that a *heap error* has occurred.

Outline

Future work

Future work

- Suggest some reduction rules (preserving system properties) to the user if some specific configurations have been found, using the analysis already explained
- Continue working on the performance issue when deciding about the reachability to a marking.

Conclusions

Conclusions

- Theoretically, algebraic analysis is more appropriate for big Petri nets than other analysis methods, like reachability graph, but in practice, some of these applications require to work with too many combinations, what makes algebraic analysis unfeasible in these cases.
- We have provided a dictionary between some properties that can be analised in a Petri net, and the equivalent equations/algorithms that the corresponding matrices (related with the Petri net) must hold.

Outline

M. D. Anna and S. Trigila.
Concurrent system analysis using petri nets: an optimized algorithm for finding net invariants.
Comput. Commun., 11(4):215–220, August 1988.

René David and Hassane Alla.
Discrete, Continuous, and Hybrid Petri Nets.
Springer, 1 edition, November 2004.

T. Murata.
Petri nets: Properties, analysis and applications.
Proceedings of the IEEE, 77(4):541–580, 1989.