

# A Petri net based representation for planning problems

**Marcos Castilho**  
**André Guedes**  
**Tiago Lima**  
**João Marynowski**  
**Razer Montaña**

Departamento de Informática,  
Federal University of Paraná,  
Curitiba, Brazil

**Luis Künzle**  
**Fabiano Silva**  
CPGEL,  
CEFET-PR,  
Curitiba, Brazil

## Introduction

In this paper we propose a Petri net based representation for planning problems. The motivation for this is that Petri nets are a formal tool useful to model and analyse domains involving true parallelism, concurrency, conflicts, and causal relations which are beyond the scope of classical planning.

In (Silva, Castilho, & Künzle 2000) we presented a way to translate the plan graph into an acyclic Petri net. This would already serve as a basis for our desired analysis on non-classical planning. However, that translation kept the same redundancies of the plan graph. It just translate propositions and actions in the plan graph to places and transitions in the Petri net.

In this first translation we didn't explore the dynamics of Petri nets. In the approach proposed in this paper we show the construction of the Petri net directly from the description of the problem. In this new structure, we give another view about the mutex relation and maintenance actions. We give details about this in section .

In Petri nets, a planning problem corresponds to a submarking reachability problem. This is known to be EXPspace-hard (Lipton 1976; Esparza & Nielsen 1994) in the general case. Fortunately, our net is an acyclic one and in this case we are in the NP-complete case (Stewart 1995), which is what we expected. Anyway, to solve the reachability problem is not straightforward and due to lack of space we refer the reader to (Rauhamaa 1990). In this paper we focus on the structure of our model.

In the next section we recall the basis of Petri nets. Then we present the construction of a Petri net directly from the description of the planning problem. Finally we present some concluding remarks.

## Petri Nets, Reachability and the Petriplan algorithm

A Petri net (Murata 1989) is a 4-tuple  $N = (P, T, Pre, Post)$  where  $P = \{p_1, p_2, \dots, p_n\}$  is a finite set of places,  $T = \{t_1, t_2, \dots, t_m\}$  is a finite set of transitions,  $Pre : P \times T \rightarrow \mathbb{N}$  is the input incidence function and  $Post : P \times T \rightarrow \mathbb{N}$  is the output incidence

function. A Petri net with a given initial marking is denoted by  $(N, M_0)$  where  $M_0 : P \rightarrow \mathbb{N}$  is the initial marking.

The Petri net dynamics is given by *firing* enabled transitions, whose occurrence corresponds to a state change of the system modelled by the net. A transition  $t$  of a Petri net  $N$  is enabled for a marking  $M$  iff  $M \geq Pre(., t)$ . This enabling condition, expressed under the form of an inequality between two vectors, is equivalent to  $\forall p \in P, M(p) \geq Pre(p, t)$ .

Only enabled transitions can be fired. If  $M$  is a marking of  $N$  enabling a transition  $t$ , and  $M'$  the marking derived by the firing of  $t$  from  $M$ , then  $M' = M + Post(., t) - Pre(., t)$ . Note that the firing of a transition  $t$  from a marking  $M$  derives a marking  $M'$ :  $M \xrightarrow{t} M'$ .

We can generalise this formula to calculate a new marking after firing a sequence  $s$  of transitions. Let us consider a matrix  $C = Post - Pre$ , called Petri net *incidence matrix*, and a vector  $\bar{s}$ , called *characteristic vector* of a firing sequence  $s$  ( $\bar{s} : T \rightarrow \mathbb{N}$ , such that  $\bar{s}(t)$  is the number of times that transition  $t$  appears in the sequence  $s$ ). The number of transitions in  $T$  defines the dimension of the vector  $\bar{s}$ . Then, firing a sequence  $s$  of transitions from  $M$ , a new marking  $M_g$  is calculated by the *fundamental equation* of  $N$ :

$$M_g = M + C \cdot \bar{s}. \quad (1)$$

We can use the fundamental equation to determine a vector  $\bar{s}$  for a given net  $N$  and two markings  $M$  and  $M_g$ . The satisfying solution must be a nonnegative integer vector, and it is only a necessary condition for  $M_g$  to be reachable from  $M$ . This condition becomes necessary and sufficient for *acyclic Petri nets*, a subclass of Petri nets that have no directed circuits (Murata 1989).

The reachability relation between markings of a firing transition can be extended, by transitivity, to the reachability of the firings of a transition sequence. Thus, in a Petri net  $N$ , it is said that the marking  $M_g$  is reachable from the marking  $M$  iff there exists a sequence of transitions  $s$  such that:  $M \xrightarrow{s} M_g$ . The reachability set of a marked Petri net  $(N, M_0)$  is the set  $\mathcal{R}(N, M_0)$  such that  $(M \in \mathcal{R}(N, M_0)) \Leftrightarrow (\exists s M_0 \xrightarrow{s} M)$ .

We call the *reachability problem* for Petri nets the problem of determining if a given marking  $M_g$  is reachable from  $M_0$ . The *sub-marking reachability problem* for a given

sub-marking  $M_s$  consists of determining if exists a marking  $M_g$  that is reachable from  $M_0$  and  $M_s \subset M_g$ , where  $M_g \in \mathcal{R}(N, M_0)$ . In (Rauhamaa 1990) we have several different techniques to solve it.

The *Petriplan* algorithm consists in two steps: first, the construction of a Petri net from the description of the planning problem; then find a sequence of transitions firings that solves the reachability problem. In the next sections we explore the construction of the net directly from the description of the problem, taking profit of the representational power of a Petri net.

## The plan net

In this section we modify the structure of our Petri net defined in (Silva, Castilho, & Künzle 2000) and define what we call the *plan net*, which is simply a Petri net obtained directly from the description of the problem exploring the representational power of Petri nets. We need however to explain two important points before showing the construction technique.

First of all, let's consider the representation of propositions. In the beginning of the construction of the net a place represents a proposition. During the process when it is found that a proposition is a precondition of more than one action, we just copy the place. It may happen that a place will be copied several times.

Now let's consider the possible inconsistencies between actions. In the plan graph this means to look for the mutex relation between action in some layer. When this is found the actions are marked as mutex, i.e., these two actions cannot be executed at the same time. This forces the copy of the entire layer to a new one using maintenance actions. In a certain sense the conflict is not completely solved, just in the "search for a solution" phase the two actions are ordered.

In our case the proposal is to have no maintenance actions. What we do is to refine the mutex relation. We relate two actions in five different ways, not only two (mutex and not mutex). Let  $x$  and  $y$  be two actions. We define the following:

- $(x \parallel y)$ : they are totally independent, that is, they may happen even in parallel. This is the "not mutex" in the plan graph sense. It may be possible to have only  $x$ , only  $y$ ,  $x$  followed (or preceded) by  $y$  and  $x$  and  $y$  in parallel;
- $(x \nabla y)$ :  $x$  has as effect the negation of some effect of  $y$ . This way  $x$  and  $y$  may occur in any order, but not in parallel;
- $(x \not\prec y)$ :  $x$  has as effect the negation of some precondition of  $y$ . So  $x$  could not occur before or in parallel with  $y$ ;
- $(x \not\prec y)$ :  $y$  has as effect the negation of some precondition of  $x$ . So  $y$  could not occur before or in parallel with  $x$ ;
- $(x \diamond y)$ :  $x \not\prec y$  and  $x \not\prec y$ . The given actions may occur just each one alone or with a third action between them.

This is an important difference between the graph and the plan net. The price for this is that we need to find out the correct kind of relation between two actions. The algorithm is based on a graph structure called *graph of static inconsistencies*, which is a graph whose nodes are actions and there

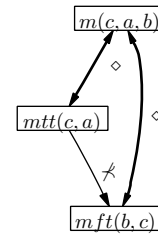


Figure 1: Graph of static inconsistencies for the first layer.

is an edge of type  $t$  linking  $x$  and  $y$  if  $x$  is related with  $y$  with respect with relation  $t$ . Observe that  $(x \diamond y)$  is the stronger case. The process of construction of this graph has the same computational cost of finding all the static mutex relations in the plan graph.

Now we are in condition to show the algorithm to construct the plan net. This process follows the idea of the construction of the plan graph. It begins with marked places representing the initial state.

We enter then in a loop looking for the places representing the final state. This loop has three phases, which are described in details below.

Phase 1: we add transitions representing all possible actions whose preconditions are already in the net. If some place is already a precondition of some other transition create a copy of this place. This copy is not needed only in the case whether the consequence of the action is the negation of that precondition been copied. This copy will be linked with the transition been added. This phase will define a layer, i.e., all possible actions that may be fired simultaneously.

Phase 2: we construct the graph of static inconsistencies for the transitions in the last generated layer (figure 1). It is constructed as we explained above. This graph will guide the construction of the *control structure of the net*. This is a Petri net containing all possible sequences of non inconsistent actions present in the last generated layer. The places here are not associated with propositions, they are just control places. We merge this structure in the net. The merge process is to include copies of the actions appearing in the control structure that are not in the original net. But we do not need to copy the places representing preconditions of the actions been copied. For example in figure 2 the action  $mft(b, c)^0$  was copied to  $mft(b, c)^1$ , but both share the same preconditions  $f(b)^1$ ,  $f(c)^0$  and  $ot(b)^0$ . At the end of this phase we have a Petri net containing all possible ways of executing the actions without any conflict in this layer. Figure 2 shows the resulting net.

We must say that the notion of layer in the Petri net is different from that in the plan graph. Here, a layer may contain actions happening in more than one instant of time, whereas in the plan graph each layer is associated with only one instant of time. Due to the process of construction based on the graph of static inconsistencies we can warrant that there is no static inconsistent sequences of actions in each branch of the net in this layer.

Phase 3: if the net contains places representing the goal state we enter phase 3, i.e., we will look for a solution. That

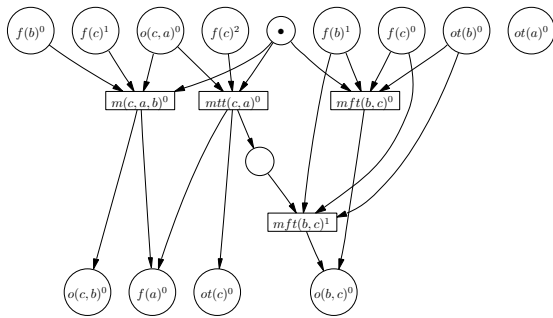


Figure 2: The first layer for Sussman anomaly with control structure.

means to find a flow in the net which puts tokens in the places representing the goal state. This is the reachability problem in Petri nets. As said, we refer to (Rauhamaa 1990) for the complexity of this problem. If such a flow exists, then it is a (possibly parallel) plan. In the other case, we return to phase 1. In our example there is no such a flow. So we must return one more time to phase 1 and 2. For lack of space we will not show the figures. Now in phase 3 the flow exists. Figure 3 the final Petri net for the Sussman anomaly. This net is a simplified version containing just the paths which reach some goal state place.

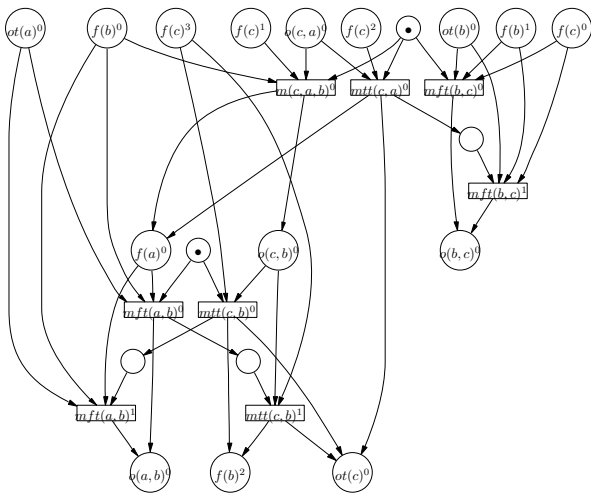


Figure 3: Final Petri net for the Sussman anomaly.

## Discussion

Relations between Petri nets and planning problems were former investigated by (Murata & Nelson 1991) and (Mieller & Fabiani 2000). The first use a general cyclic predicate-transition Petri net. The problem is that the necessary and sufficient condition of equation 1 is broken, and the only way to solve the reachability problem is to use the reachability graph, which leads to an enumerative search for a solution.

The second approach defines a cyclic coloured Petri net,

in which each place corresponds to a logical predicate describing actions preconditions or effects. The operators instantiation is made by token colours. The theoretical model obtained for the resulting planning problem is in fact more compact than ours, but it presents the same problem of exhaustive search, as in (Murata & Nelson 1991).

In our approach, however, we have a simpler acyclic place-transition Petri net, with necessary and sufficient conditions to use the equation 1 to find a solution to the planning problem. This paper modifies our first presentation of the *Petriplan* algorithm (Silva, Castilho, & Künzle 2000) by taking profit of the dynamics of the Petri net thus reducing the structure.

Finally, the method proposed in this paper permits to construct a Petri net representation of the planning problem. As others methods, we can find a solution to the planning problem, in our case using reachability algorithms. The classical way is to start an exhaustive search, just as *Graphplan* does. However, as we have an acyclic Petri net, the matrix representation of the fundamental equation can be viewed as a constraint satisfaction problem, which can be solved using several methods, as integer programming, SAT, among others.

## References

- Esparza, J., and Nielsen, M. 1994. Decidability issues for Petri nets - a survey. *Bulletin of the European Association for Theoretical Computer Science* 52:245–262.
- Lipton, R. J. 1976. The reachability problem requires exponential space. Technical report, Dept of Computer Science, Yale University. research report 62.
- Mieller, Y., and Fabiani, P. 2000. Planning with Petri nets. In *Proc. of RJCIA-2000*.
- Murata, T., and Nelson, P. 1991. A predicate-transition net model for multiple agent planning. *Information Sciences* 57-58:361–384.
- Murata, T. 1989. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* 77(4):541–580.
- Rauhamaa, M. 1990. A comparative study of methods for efficient reachability analysis. Technical Report A 14, Digital Systems Laboratory, Helsinki University of Technology. <http://citeseer.nj.nec.com/245545.html>.
- Silva, F.; Castilho, M.; and Künzle, L. 2000. Petriplan: a new algorithm for plan generation (preliminary report). In *Proc. of IBERAMIA/SBIA-2000*, 86–95. Springer-Verlag.
- Stewart, I. A. 1995. Reachability in some classes of acyclic Petri nets. *Fundamenta Informaticae* 23(1).