Concurrent Planning using Petri Net Unfoldings

Sarah Hickmott
National ICT Australia and
School of Electrical and Electronic Engineering University of Adelaide, Australia

Abstract

Those in the planning community previously exposed to the construct of Petri nets, have probably recognised a connection between the world modelled by these nets and a concurrent planning domain. Work to date however has failed to sufficiently capture and motivate the possible benefits of developing and utilising this connection, so the area has remained relatively undeveloped. We believe the factored state representation and concurrency semantics of Petri nets are so closely related to that of concurrent planning domains that cross cultivation between the areas will give theoretical insight and lead to the development of computationally efficient algorithms for concurrent planning. Our research seeks to develop the theory required to exploit the connection and create models and algorithms for optimally solving deterministic and probabilistic concurrent planning problems using low level Petri nets. Our approach casts the planning problem as a Petri net reachability problem. Unfolding is an attractive method for reachbility analysis because it utilises and maintains the concurrency and factored state structure of the Petri net. We capitalise the unfolding process for planning by guiding it with heuristics.

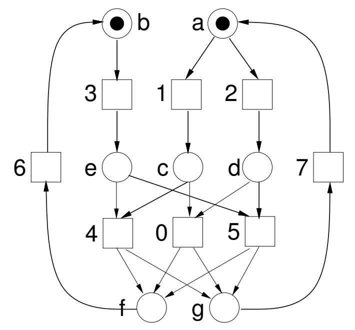
Petri nets are traditionally used for modeling and analysis of distributed systems. They can be used to exploit the structure of a planning domain in two beneficial ways. Firstly Petri nets, like STRIPS and PDDL operators, provide a non-flat representation of transition systems. They avoid explicitly enumerating the state space as it is implicit in their representation of variable -action relationships. This can be utilised in computation. Secondly Petri nets specifically represent concurrency and causal relations between actions. Consequently it is possible to avoid enforcing a total order on actions; this can have not only computational advantages via reduction of the policy space but also allows us to generate partially ordered plans.

We propose a translation from deterministic and probabilistic planning problems to Place-transition (PT) Petri nets. We then unfold the net in a specific manner. The unfolding of a Petri net maintains a partial ordering of actions based on causality, allowing us to search for an optimal solution plan without considering the unnecessary interleaving of actions or enumerating the state space entirely. For the deterministic case, we adapt the Esparza-Vogler-Romer (EVR) unfolding algorithm so that either the minimum cost partial plan is found during the unfolding procedure, or we identify that the goal is not achievable. We capitalise the unfolding by guiding it with planning heuristic. Note we are not restricted to unit-cost actions. For the probabilistic case, we unfold the Petri net up to a finite horizon. We propose a Bellman-like equation which then uses the unfolded net to determine the partial plan with the maximum probability of success, or expected reward, within the finite horizon. This later case does not require that all rewards be positive, and thus could be used for over-subscription planning.

Translation of a Planning Problem to a Petri net

A Place Transition (PT) net is a low level Petri net. A PT-net is a 5-tuple $PN = (P, T, F, W, M_0)$ where P and T are finite sets of places and transitions respectively; $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation; $W : F \rightarrow 1, 2..$ is the weight function; M_0 is the the initial marking; $P \cap T = \emptyset$ and $P \cup T \neq 0$. Figure 1 shows a simple PT net. The marking M of a Petri net is the mapping of tokens to places, and represents the state of the modelled world. The preset of a node x in the net, ${}^\bullet x$, is the set $\{y \in P \cup T | W(y, x) \geq 1\}$. The postset of a node, $x^\bullet$, is the set $\{y \in P \cup T | W(x, y) \geq 1\}$. A transition t is enabled if each of its input places p is marked with the weight of the arc connecting p to t. A particular marking M enables a transition t if $W(p, t) \leq M(p)\ \forall p \in P$. The occurrence of a transition absorbs the tokens in its input places and produces tokens in its output places thus moving the net from M to the new marking $M'(p) = M(p) - W(p, t) + W(t, p)\ \forall p \in P$. This corresponds to a state transition of the modelled system. A set of transitions T' is concurrently enabled at the marking M if it is possible for all $t \in T'$ to occur at once, viz. $\sum_{t \in T'} W(p, t) \leq M(p) \forall p \in P$. A more detailed review of Petri nets can be found in (Murata 1989).

A planning problem is a quadruple $\langle A, I, O, G \rangle$ where A is a set of state variables, $I : A \rightarrow \{0, 1\}$ is a state, O is a set of STRIPS operators, and G is a set of goal literals (Ghallab, Nau, & P.Traverso 2004). The set of literals over A is $L = \cup\{\neg a | a \in A\}$. The complement $\bar{l}$ of a literal $l \in L$ is defined by $\bar{a} = \neg a$ and $\bar{\neg a} = a$ for $a \in A$. A STRIPS operator $\langle p, e \rangle$ is 1-safe if $\{\neg l | l \in e\} \subseteq p$. A STRIPS operator $\langle p, e \rangle$ has a positive precondition if $p \subseteq A$. The first part of our translation involves mapping the planning problem to an equivalent one where every operator is 1-safe one, and there are no negative preconditions. 1-safety is established



Places are represented by circles, transitions by squares and token by black dots. Firing transitions 2 and 3 (in any order or concurrently) followed by firing transition 5, would result in a single token each in places f and g.

Figure 1: Example of a Petri net.

by replacing every operator by several 1-safe ones; we define $S(o)$ as set of operators obtained from some $o \in O$. Negative preconditions are eliminated by replacing each $\neg a$ by a corresponding positive precondition $\hat{a}$ and forcing $\hat{a}$ and a to always have opposite values. The reasons for this translation become clear soon.

Given a deterministic planning problem $R = \langle A, I, O, G\rangle$ we create a PT $pnet(R) = \langle P, T, F, W, M_o\rangle$ such that:

- the places are $P = A \cup \hat{A}$,
- the transitions are $T = \{S(o)|o \in O\}$,
- the set F of arcs is obtained from $t_0 = \langle p, e\rangle \in T$ as

$$\begin{aligned}&p \times \{t_0\}\\ &\cup\{\langle t_o, a\rangle | a \in A \cup \hat{A}, a \in p, \neg a \notin e\}\\ &\cup\{\langle t_o, a\rangle | a \in A, a \in e\}\\ &\cup\{\langle t_o, a\rangle | a \in A, \neg a \in e\}\end{aligned}$$

- $W(f) = 1$ for all arcs $f \in F$,
- for all $a \in A$, $M_0(a) = 1$ iff $I(a) = 1$ and $M_0(\hat{a}) = 1$ iff $I(a) = 0$, and for all $a \in A \cup \hat{A}$, $M_0(a) = 0$ or $M_0(a) = 1$.

The 1-safety of the STRIPS operators has allowed us to create 1-safe Petri nets, meaning it is not possible for more than one token to exist in a place. Without 1-safeness it would become complicated to maintain consistency in the net, as one would have to consider the semantics of multiple tokens in a place: if the respective literal becomes false all these tokens must be removed.

Petri net Unfolding

Unfolding is a method for reachability analysis which exploits and preserves the factored state representation and concurrency information in the Petri net. In the planning context this confers the ability to reason about partially ordered sets of actions directly, without having to consider their interleavings. It also enables the recognition and separate resolution of independent subproblems. During the planning process we can reason about the actions and cost required to assert some subset of state variables, and combine this information with that for another (thus-far) independent subset, at a future point when the two sets are no longer independent. The benefit of this 'divide-and-conquer' approach depends on the level of concurrency in the domain.

The unfolding of a PT-net $N = (P, T, F, W, M_0)$ produces an occurrence net $ON = (B, E, F')$, whose node sets are conditions B and events E. These nodes represent particular occurrences of the places and transitions, respectively, in possible runs of the original net from the initial marking. The unfolding achieves this by eliminating cycles and backward conflicts. Backward conflict is the case when two transitions output to the same place; by eliminating this we know exactly which transitions are involved in a particular marking. In the context of planning, this means that we know the exact set of actions that, when executed, lead to a state variable reaching a certain value at some point in the plan. The labelling function φ is a homomorphism from ON to N.

The main theoretical notions required to understand unfolding are that of a configuration and local configuration of an event. A configuration represents a possible partial run of the net. It is any set of events C such that: C is causally closed, e $\in C \Rightarrow e' \in C\ \forall e' \leq e$; and C contains no forward conflict, $^\bullet e_1 \cap {}^\bullet e_2 = 0\ \forall\ e_1, e_2 \in C,\ e_1 \neq e_2$. We can think about a configuration as a partially ordered plan. The local configuration of an event e, denoted $[e]$ is the minimal configuration containing event e. Consequently if we introduce a 'goal' transition t_g whose predecessors correspond to the set of goal literals G, then a solution plan is any partially ordered set of operators [e] such that $\varphi(e) = t_g$. A configuration C can be associated with a marking Mark(C) of the original net by identifying those conditions whose tokens are produced but not consumed after firing the events in C starting from the initial marking: Mark(C) $= \varphi((M_0 \cup C^\bullet)\backslash {}^\bullet C)$, where $C^\bullet = \{e^\bullet | e \in C\}$ and $^\bullet C = \{^\bullet e | e \in C\}$.

The unfolding process involves identifying which transitions are enabled by conditions currently in the occurence net that can be simultaneously marked. These transitions are referred to as the possible events. A new instance of each is added to occurence net, as are instances of the places in each of their postsets. The question of whether a set of conditions can be simultaneously marked is answered by determining whether the union of the local configurations of their presets forms a configuration. Figure 2 shows an example of unfolding.

Finite Complete Prefix of Unfolded net

In most cases, the unfolding β of a Petri-net is infinite. For this reason, we seek a complete finite prefix β' of β, one which contains as much information as β.

The key to obtaining a complete finite prefix is to identify those events at which we can cease unfolding without loss of information. Such events are referred to as *cut-off events* and are defined in terms of an *adequate order* on configurations (McMillan 1992; Esparza, Römer, & Vogler 2002):

Definition 1 *A partial order $\prec$ on the finite configurations of a branching process is an adequate order if*

1. *$\prec$ is well founded*
2. *$\prec$ refines $\subset$: $C_1 \subset C_2 \Rightarrow C_1 \prec C_2$*

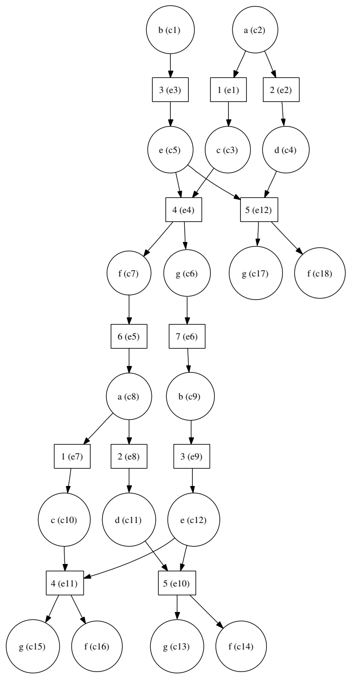


Figure 2: Prefix of an unfolding of the Petri net in Figure 1.

3. *$\prec$ is preserved by finite extensions: if $C_1 \prec C_2$ and Mark(C_1) = Mark(C_2) then $C_1 \oplus E \prec C_2 \oplus I_1^2(E)$, where $I_1^2(E)$ is an isomorphism mapping the finite extension E of C_1 onto the extension of C_2.*

Without loss of information, we can cease unfolding from an event e, if e takes the net to a marking which can be caused by some other event e' such that $[e'] \prec [e]$. This is because the events (and thus markings) which proceed from e will also proceed from e'. Relevant proofs can be found in (Esparza, Römer, & Vogler 2002):

MOLE (*http://www.fmi.uni-stuttgart.de/szs/tools/mole/*), is a free-ware unfolder which can be used for 1-safe PT-nets. It uses an adequate order on configurations, $\prec$, which is based primarily on comparing their cardinality. The prefix shown in Figure 2 is the complete finite prefix MOLE returns for the net in Figure 1.

Deterministic Concurrent Planning

Once the problem is translated to a PT-net, it is easy to let MOLE produce a partially ordered plan for that problem. It suffices to augment the STRIPS operator set with a dummy operator whose precondition is the goal, and to require mole to stop whenever an event labelled with the corresponding transition is added to the occurence net. The local configuration of this event is a partially ordered plan for the problem. Further, owing to the fact that MOLE orders events by increasing local configuration cardinality, this plan is minimal in the number of actions.

The cardinality-based ordering relation used by MOLE has a serious drawback for planning however, as it leads MOLE to perform a breadth-first search. If we were to swap the ordering to prefer events with larger local configurations to those with smaller ones, we would force MOLE into a depth-first search. However, since the resulting ordering is not adequate, there is no guarantee that the resulting planner would be complete. Checking for loops is not as straightforward in the unfolding framework as it is in a state-space search framework, since the markings of local configurations only have a partial view of the state. Indeed, the main purpose of an adequate order is to implement a form of loop detection. A natural idea is to change the ordering to provide MOLE with better guidance towards the goal, while preserving, and even generalising from the restricted notion of optimality currently in place. This rejoins the work on directed model-checking pionneered by Edelkamp et al. (Edelkamp, Lluch-Lafuente, & Leue 2001).

We propose that given an arbitrary *monotonic* heuristic, it is possible to build an adequate order which implements A*, letting the heuristic guide the unfolding towards optimal plans. Monotonic heuristics which, like h^m (Haslum & Geffner 2000), can be automatically generated from a planning problem description, are equally easily generated from PT-nets. In planning terms, let cost(o) be the (positive) cost of operator o, and res(o, s) be the result of applying o in state s. A Heuristic h (such that $h(s) \geq 0$ everywhere and $h(s) = 0$ at goal states) is monotonic iff $h(s) \leq h(\text{res}(o, s)) + \text{cost}(o)$ for all non-goal states s and operators o applicable in s. These definitions easily transfer to the PT-net case, by identifying each operator with the corresponding transition and considering a set of places P as the state in which all state variables but those in P are false. We define the following ordering on configurations:

Definition 2 ($\prec_h$) *Let h be a monotonic heuristic as defined above. For a configuration C, define $g(C) = \sum_{e \in C} cost(\varphi(e))$, and $f(C) = g(c) + h(\text{Mark}(C))$. Define $C \prec_h C'$ if and only if $f(C) < f(C')$ or $f(C) = f(C')$ and $|C| < |C'|$.*

Proof that this order is adequate will not be shown here. When running MOLE with this ordering for some monotonic heuristic h, we obtain a planner which generates partially ordered plans with the smallest total action cost. As far as we are aware, only the HSP* family of existing planners routinely optimise this metric (Haslum, Bonet, & Geffner 2005). In contrast, most state of the art planners optimise parallel plan length. It is possible our approach could be modified for concurrent temporal planning, but the full implications of this have not yet been considered.

Experimental Results

Our translation from propositional STRIPS operators to PT-nets is implemented in Standard ML within a program called Petrify. Petrify actually parses a large subset of PPDDL (Younes *et al.* 2005), and handles non-grounded domains, conditional, and probabilistic effects. We modified MOLE to implement a variety of search strategies and heuristics defined by their respective ordering relations. In Figure 3, we present results for PIPESWORLD and AIRPORT instances, for some of the variants as they provide a good illustration of the benefits and problems with our current implementation. Namely we show results for $h(s) = 0$, $h(s) = h^1_{max}$

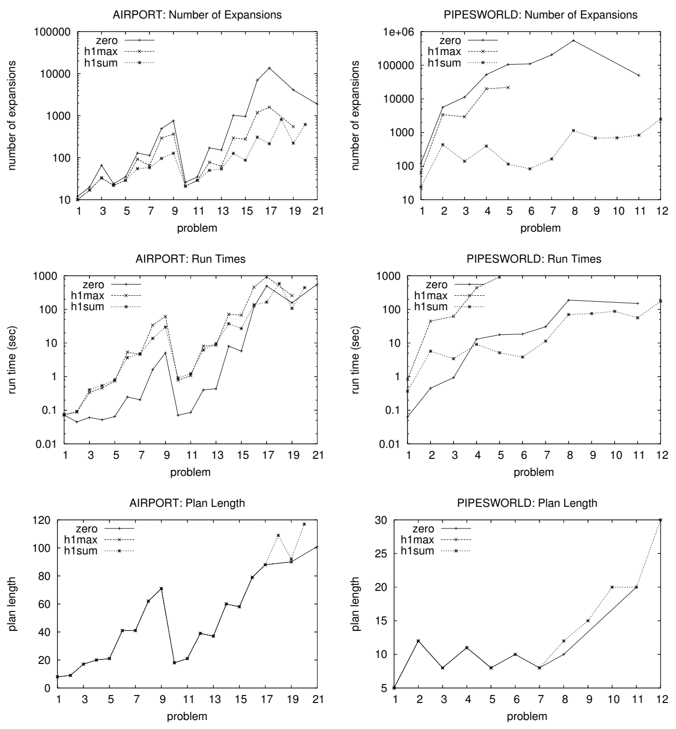


Figure 3: Experimental Result on PIPESWORLD and AIRPORT Instances

and $h(s) = h^1_{sum}$, where the later is the same as h^1_{max} but using the sum instead of the maximum. Out of these, only $\prec_0$ and $\prec_{h^1_{max}}$ are adequate orders and thus guarantee completeness.

As can be seen from the figures, the number of node expansions performed by MOLE significantly decreases (note the log scale) when switching from $h = 0$, to the monotonic h^1_{max} heuristic, and then to the non-admissible h^1_{sum}. This is particularly visible in PIPESWORLD where h^1_{max} and h^1_{sum} respectively expand over an order of magnitude and over three orders of magnitude fewer nodes than $h = 0$. However, this improvement in number of expansions does not carry over to run time. Without heuristic, our AIRPORT run times are as good as, and in a number of cases better than those obtained by the competition optimal planners, except SATPLAN. The h^1_{max} heuristic is always slower than breadth-first and h^1_{sum} only start yielding run-time gains when the improvement in number of expansions reaches two orders of magnitude. This is because, in a forward search, h^1 heuristics need to be recomputed at each expansion, and this computation has a complexity quadratic in the number of nodes of the PT-net. We expect to see a significant improvement in run time by switching to other automatically generated heuristics which can be pre-computed once and efficiently looked up during the search. Promising candidates include Pattern Database heuristics (Edelkamp 2002; Haslum, Bonet, & Geffner 2005).

Probabilistic Concurrent Planning

The translation for the probabilistic case is similar but somewhat more complicated and will not be formalised here. For probabilistic concurrent planning, we are currently focused on optimising the maximum expected reward, given a finite horizon. The Petri net is first unfolded completely, without generating any cut-off points, until the decided horizon. We propose a Bellman-like equation which uses the unfolding to answer the question: given this set of conditions are marked, what is the maximum expected reward from here and what event must be chosen to achieve this? This will not be discussed further here, due to space constraints.

Future Work

Experimental results are not presently entirely conclusive. They are only competitive with state of the art planners in some of the domains examined. Our immediate agenda is to implement pattern databases heuritics. Furthermore we will consider different translations to PT-nets and to higher level nets.

Acknowledgements

I would like to acknowledge Sylvie Thiébaux, Jussi Rintanen and Langford White for their ongoing contributions to this research.

References

Edelkamp, S.; Lluch-Lafuente, A.; and Leue, S. 2001. Directed explicit model checking with hsf-spin. In *SPIN*, 57–79.

Edelkamp, S. 2002. Symbolic pattern databases in heuristic search planning. In *AIPS*, 274–283.

Esparza, J.; Römer, S.; and Vogler, W. 2002. An improvement of mcmillan's unfolding algorithm. *Formal Methods in System Design* 20(3):285–310.

Ghallab, M.; Nau, D.; and P.Traverso. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann Publishers.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *AIPS*, 140–149.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *AAAI*, 1163–1168.

McMillan, K. L. 1992. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *CAV*, 164–177.

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

Younes, H. L. S.; Littman, M. L.; Weissman, D.; ; and Asmuth, J. 2005. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research* 24:851–887.