

Abstract

The problem of planning in a world can be seen as the problem of determining a sequence of world changes allowing to reach a goal state from a given initial state. Each world change is caused by an action. We present here a planning system which is constructed on an action formalism. Actions are pairs of formulae (precondition, result). The precondition describes what has to be true in a state for an action to be performed; the result describes what will be true in a state after the execution of the action. Actions occur in worlds which are also determined by general laws which remain always true throughout the changes. The general laws indicate which are the possible states for a world. They can be obtained from the implicants of the general laws. Given a set of actions and general laws, we show how to obtain an action graph describing all possible changes. The problem of planning is then the problem of finding a path in this graph relying two states the specifications of which are given. We describe an implementation of the system using a tableaux prover.

1 Introduction

In this article, we present a planning system which is based on a logical theory of actions. This theory specifies an action by a pair of formulae (precondition, result). The precondition describes which facts have to hold in a state of the world in order to perform the action and the result describes which facts will be true after the performance of the action. When the action is executed, its results will be true in the following state. The world is characterized by general laws which have to be true in every state. Each action is given as a pair of formulae (preconditions, results). Moreover, every action is associated with a set of persistencies, i.e. literals that have to persist as the action is executed. The planning problem is then the following: given two states s and s' of the world. s being the actual state and s' the goal state, find a sequence of actions $a_1, a_2, a_3, \dots, a_n$ such that there is a sequence of states $s_1, s_2, s_3, \dots, s_{n-1}, s'$ and a_1 can be performed in s yielding s_1 , a_2 can be performed in s_1 yielding s_2 , ... and a_n can be performed in s_{n-1} yielding s' . The states of the world can be seen as models of the general laws. So, we have to solve the problem of model generation. Models of a formula correspond to implicants of this formula: every implicant represents a set of models (the models validating it). We have used an analytic tableaux prover to obtain implicants.

The system we propose is organized as follows: given the general laws, possible states are constructed as implicants of these laws. An action arc for an action (C, R) is set between two states s, s' when C is true in s and R and the persistencies are true in s' and when s and s' changes as few as possible otherwise. Given a set of actions, the action graph is thus constructed over

these implicants. The planning problem consists then in finding the shortest path in this graph which leads from an initial state to a goal state.

This article is organized as follows: in the next section, we present the underlying action theory. The third section introduces analytic tableaux and their application to implicant generation. Section 4 describes how the action graph is constructed from the general laws and the action specifications. Our system is implemented and has been run with a number of examples the results of which are presented in section 5.

Throughout this paper, we will use the following notations.

Notations

We will use first-order language L as usual. We denote by Lit^+ the set of atoms and by Lit the set of literals of L .

An interpretation is a truth-valued function on Lit^+ which can be extended to arbitrary formulae according to classical truth tables.

According to general use in action theories, we call *state* any interpretation which validates the general laws (i.e. any model of the general laws). Frequently, we denote an interpretation as the set of the literals it assigns true. Given a set of formulae F , $Mod(F)$ is the set of models of F .

2 Action theory

As we have already explained, the problem is that most facts do not change as an action is performed. On the other hand, some facts change without being among the direct consequences of an action. In our action theory, we distinguish three kinds of behaviour of facts according to persistence and change:

1. Facts that have to be true after the action execution. These facts will appear as direct action consequences in action specifications.
2. Facts that have to persist as an action is performed. These facts, which will also be called *strong persistences* will be associated to every action. Typically strong persistencies include facts which are not among the direct results of an action, but are sufficiently “close” to its result.
3. Facts that will persist as far as this is possible. These facts will persist as changes are minimized.

Definition 1 (Action theory) *An action theory AT is a triplet (W, AC, sp) where*

1. $W \text{ subset } L$ is the set of general laws supposed to be true in every state of the world.
2. $AC = \{(C, R) : C \in L, R \in L\}$ is a set of actions, where C is called precondition and R is called result
3. sp is a function from AC to the set of subsets of Lit which assigns to every action a the set of literals which have to persist as a is performed. $sp(a)$ is called the set of strong persistences of a .

The general laws are formulas which have to be true in every state of the world. They determine what is possible in a world, or what are the possible situations. We will use the concept of situation of a world which is a syntactical concept and designs any formula which is “possible” with respect to the general laws of a world.

Definition 2 (Situation of a world) *Given a world W , a situation for W is any first-order formula F consistent with W , i. e. any first-order formula F such that there is no finite subset $\{A_1, A_2, \dots, A_n\}$ of W such that $\vdash \neg A_1 \vee \dots \vee \neg A_n \vee \neg F$*

An action can possibly be executed in a situation, when its preconditions can be derived from the situation under the general laws. The resulting situation is then determined by the set of models of the action result “close” to the models of the original situation. Our definition of “action result” is an update type definition (see [?][?]), i.e. depends on models and minimizes change according to models. Our definition can be seen as a variant of Winslett’s possible models approach [Win88] (PMA), where the difference concerns the possibility to take into account strong persistencies.

Definition 3 (Possible action) *Given an action theory $AT = (W, AC, sp)$, $a = (C, R) \in AC$ and a situation A of W , a is possible in A iff $W \cup \{A\} \vdash C$*

The set of resulting states after execution of an action is defined for every model of a world situation and is defined to be the set of models of the result of the action preserving persistencies and “closest” to the models of the situation.

Definition 4 (Execution result) *Given an action theory $AT = (W, AC, sp)$, $a = (C, R) \in AC$, executable in a situation A of W , M a model of A and W , the result of the execution of a in M is defined by $Res(M, a) = \{M' :$*

1. $M' \in Mod(R) \cap Mod(W)$

2. $sp(a)M \subset M'$
3. *there is no $M'' \in Mod(R) \cap Mod(W)$ such that $M \div M'' \subset M \div M'$*

Example 1 *Let be $AT = (W, AC, sp)$ with*

1. $W = \{l \leftrightarrow (u_1 \leftrightarrow u_2)\}$.
2. $AC = \{a\}$ where $a = (u_1, \neg u_1)$.
3. $sp(a) = \{u_2\}$

W contains one law describing a lamp l commanded by two switches u_1 and u_2 ; a is the action “toggle1(u_1)”, which changes u_1 to $\neg u_1$. As a occurs, clearly u_2 should not change. Therefore, we include u_2 into the persistencies of a . In the pure PMA (without persistencies the model $\{l, u_1, u_2\}$ of W has two results from the performance of a , namely $\{\neg l, \neg u_1, u_2\}$ and $\{l, \neg u_1, \neg u_2\}$. Clearly, only the first result is desired because in the second one, u_2 has changed although it has no relationship whatsoever with the action a or its result $\neg u_1$.

In the circumscriptive formulation the theory has also two resulting states. In order to prevent the unintuitive one, Lifschitz introduces the concept of frame fluents [?] and for this example, u_1 and u_2 are frame fluents. The difference with our approach is that we think that frame fluents should depend on actions whereas for Lifschitz, they are global for a whole theory.

3 Analytic tableaux

Given that actions may be performed between states, we turn now to the issue of determining the states between which actions may occur. In order to do that, several methods may be used. We can produce states by deductions of actions consequences according to general laws (Planning resolution with theorem prover), or produce all the states satisfying the general laws and to study the actions performances between them (cf. Hertzberg, Thiebaut). But states, which are costly to generate, often contain informations not needed to apply actions. For instance, one does not need to know if a book is on the desk if one wants to move himself. Therefore, it will be interesting to produce some reduced forms which represent states satisfying the general laws and which can be simply and efficiently used to determine when actions may be performed.

Analytic tableaux generate a set of implicant sets for a formula corresponding to its disjunctive normal form. Every model is thus represented. We have used analytic tableaux because we think they produce model representations in a very natural way.

3.1 Analytic tableaux and implicants

An analytic tableaux prover tries to prove a formula f by deriving a contradiction from $\neg f$. It constructs for $\neg f$ a tree each branch of which is a set of formulas. It can be viewed as a set of sets of literals corresponding to its disjunctive normal form.

Definition 5 (Closed tableau) *A set of literals is closed if it contains two opposite literals, it is open otherwise. A tableau is closed if each of its branches is closed. A tableau is open if it is not closed.*

Theorem 1 (Fundamental property of tableaux) *Let f be a formula,*

$$\vdash f \text{ iff } TP(\{\neg f\}) \text{ is closed}$$

Proof 1 *cf. [Smu68]*

Another interest emerges from the properties of the open branches of a formula's derivation.

Proposition 1 (Satisfiability of a formula) *Let F be a formula, F is satisfiable iff $TP(F)$ is open.*

Proposition 2 (Implicants of a formula) *Let F be a formula. The conjunction f of the literals of an open branch of $TP(F)$ is called implicant of F and has the property*

$$f \vdash F$$

Furthermore, let $I = f_1 \vee \dots \vee f_n$ be the disjunction of all open branches of $TP(F)$ (implicants of F),

$$I \vDash F$$

Hence,

$$\text{Mod}(F) = \text{Mod}(I)$$

Proof 2 *cf. satisfiability of a formula [Fit90]*

We have used an analytic tableaux theorem prover TS which avoids subsumed branches to obtain such conjunctive forms representing reduced forms of states.

3.2 Pre-treatment of actions

Actions are described by a couple of formulas (precondition, result) usually costly to manipulate. We can use analytic tableaux to derive the precondition and the result. So, we obtain a disjunction of their implicants which can be easily compared to the implicants previously obtained. Each action will be replaced by a group of actions which couples are combination of implicants of precondition with implicants of result.

FORMULA:

Note that this way of actions pre-treatment do not really allow to cope with non-determinism. In fact, they will be replaced by a group of determinist actions.

3.3 Replacing formulas for graph construct

All formulas have been derived by an analytic tableaux prover to obtain their implicants. We show that all possible actions transitions between states can be constructed as transitions between implicants by starting from the tableaux elements. According to definition 3 (possible action), an action (C, R) is possible in a situation A in the world W , whenever $W \cup \{A\} \vdash C$. Recall that a situation of W is any formula consistent with W . Here, we show that this property can be recovered in a general way at the level of implicants of W .

Theorem 2 *Let F be a formula and W a set of formulas of L , $\exists A \in L$ such that $W \cup \{A\} \vdash F$ iff $\exists w \in TS(W)$ and $f \in TS(F)$ such that $\{w \cup f\}$ is open.*

Proof 3 *Let be $W \cup \{A\} \vdash F$. Since $W \wedge A$ is consistent, we do not have $W \cup \{A\} \vdash \neg F$. Hence, $TS(W) \otimes TS(A) \otimes TS(F)$ is not closed, i.e. there is at least one $w \in TS(W)$, one $a \in TS(A)$ and one $f \in TS(F)$ such that $\{w \cup a \cup f\}$ is open. Therefore $\{w \cup f\}$ is open.*

On the other hand, let be $w \in TS(W)$ and $f \in TS(F)$ such that $\{w \cup f\}$ is open. Since $\{w \cup f\} \in TS(W) \otimes TS(F)$, $TS(W) \otimes TS(F)$ cannot be closed. Hence, $W \not\vdash \neg F$. But this is equivalent to say that F is a "state" consistent with W . Since $W \wedge \{F\} \vdash F$, F is the desired state.

We apply this theorem to the action graph construction.

Corollary 1 (Possible action transition and implicants)

Let $a(C, R) \in AC$, a is possible w.r.t W iff $\exists c \in TS(C), r \in TS(R)$, $w \in TS(W)$ and $w' \in TS(W)$ such that $\{w \cup c\}$ and $\{w' \cup r\}$ are open.

Therefore, we can determine if an action transition is possible by using implicants of formulas instead these formulas.

4 Planning

We turn now to the ways of finding optimal solution of a planning problem. First, we will detail a complete and correct method of building a graph which nodes are states of the problem and arcs are the executable actions between these states. Then, this graph will be examined to determine the shortest path from the initial state to the goal state representing the expected solution. This method is costly enough, but it has proven good performances in comparison with modern planners. A more efficient method which constructs the graph while searching the shortest path will be showed in section 6.

4.1 Close implicants

The definition 4 lead us to determine the set of result states of an action performance which are close to the initial situation. However, this definition is applied on models but not on implicants. Hence, we have to introduce a new operator which determines the difference between implicants.

Definition 6 (Implicants difference) *Let w and w' be two sets of literals of Lit . $w - w'$ is the implicant difference defined in the following way*

$$w - w' = \{l \in Lit : l \text{ positive literal such that } (l \in w \text{ and } \neg l \in w') \text{ or } (\neg l \in w \text{ and } l \in w')\}$$

Theorem 3 (Models) [?, Sch93]kljl

Compute w_p and w_r

...

4.2 Graph computation

At the begining, the states of our graph are the implicants of W . We already know to determine which actions may be executed from a given state and which are the results of these actions performances. This treatment has the following form:

For all $a(C, R) \in AC$

For all $w \in TS(W)/\{w \cup c\}$ is open, $c \in TS(C)$

Compute $Res(w, a) =$ set of possible result states

```

    Compute  $w_p$ 
    Compute  $w_r$ 
    Action between  $w_p$  and  $w_r$ 
  EndFor
EndFor

```

To cope with the frame problem only while computing the graph, we will represent, for each executable action, the states w_p and w_r as nodes of the graph and the oriented arc from w_p to w_r as the corresponding action.

4.2.1 Carrying on informations

To avoid adding redundant informations, we will replace w and w_{perf} by two equivalent set of states. These states, which will be called in the following sons of w and w_{perf} respectively, are defined by:

$$\begin{aligned}
 Sons(w) &= \{\{w_p\}, \{\{w\} \cup \{l_1, \dots, l_{i-1}\} \cup \{\neg l_i\}\}, 1 \leq i \leq n\} \\
 &\quad \text{with } w_p = \{w\} \cup \{l_1, \dots, l_n\} \\
 \text{and } Sons(w_{perf}) &= \{\{w_r\}, \{\{w_{perf}\} \cup \{l_1, \dots, l_{j-1}\} \cup \{\neg l_j\}\}, 1 \leq j \leq m\} \\
 &\quad \text{with } w_r = \{w_{perf}\} \cup \{l_1, \dots, l_m\}
 \end{aligned}$$

4.2.2 The Subsumption problem

4.2.3 Adding action

4.2.4 The fix point

4.2.5 Search of the shortest path

5 System description and examples

6 An overview of a more efficient planner

7 Conclusion

References

- [Fit90] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, 1990.
- [Smu68] R. M. Smullyan. *First Order Logic*. Springer-Verlag, 1968.
- [Win88] Marianne Winslett. Sometimes, updates are circumscription. In *Proceedings of the 7th American National Conference on Artificial Intelligence (AAAI-88)*, pages 89–93. Morgan Kaufmann, 1988.