Irrelevant Actions in Plan Generation (Extended Abstract)

Vladimir Lifschitz and Wanwan Ren

Department of Computer Sciences, University of Texas at Austin
Austin, TX 78712, USA
{vl,rww6}@cs.utexas.edu

Abstract. In a planning problem, some of the available actions may be irrelevant for achieving the given goal. We make this idea precise by defining, for an action description in language $\mathcal{C}+$, when a subset of its signature is "isolated." If all fluent constants mentioned in the goal of a planning problem belong to an isolated set σ then any valid solution to the problem will remain valid if we drop from it all actions that do not belong to σ. Furthermore, this more economical plan will remain valid if we drop all assumptions about the initial values of the fluent constants that do not belong to σ. Identifying isolated sets can be used to simplify the statement of a planning problem.

1 Introduction

In a planning problem, some actions can be of no use for achieving the given goal. John McCarthy [1998] introduced, for instance, an elaboration of the familiar Missionaries and Cannibals puzzle in which the missionaries and the cannibals have hats, and the hats can be exchanged among them. He observed that exchanging hats is of no use if the goal is to cross the river. There exist valid solutions, of course, in which the missionaries and the cannibals execute the action of exchanging hats at various points in time, but these actions can be always dropped from a solution. Why is it obvious, McCarthy asks, that exchanging hats is of no use?

Clearly, the extended Missionaries and Cannibals domain consists of two parts, and one of them is, intuitively, "isolated" from the other. Part 1 involves the locations of objects (missionaries, cannibals and the boat) and the action of crossing the river. It is isolated from Part 2 that involves the hats and the action of exchanging them. Since the goal (everybody is at the other bank) does not refer to Part 2, the actions from Part 2 would not help us achieve it. Interestingly, Part 2 is apparently not "isolated" from Part 1. Indeed, if initially two missionaries are at different banks and the goal is for one of them to wear the other's hat, it cannot be achieved without crossing the river.

In this paper, we show how to make the idea of an isolated part precise. This is important for several reasons. First, some computational methods used for plan generation (for instance, satisfiability planning [Kautz and Selman, 1992]) may

Notation: m, m' range over $\{M_1, M_2, M_3\}$; l ranges over $\{Bank_1, Bank_2\}$.

Simple fluent constants:	Domains:
$Loc(m)$, $Loc(Boat)$	$\{Bank_1, Bank_2\}$

Boolean action constants: $Cross(m)$

Causal laws:

$Cross(m)$ **causes** $Loc(m) = l \wedge Loc(Boat) = l$ **if** $Loc(m) \neq l$

nonexecutable $Cross(m)$ **if** $Loc(m) \neq Loc(Boat)$
nonexecutable $Cross(M_1) \wedge Cross(M_2) \wedge Cross(M_3)$

exogenous c	for every action constant c
inertial c	for every fluent constant c

Fig. 1. Missionaries crossing

actually produce plans that contain irrelevant actions.[1] A method for identifying such actions can help us make these plans more economical. Second, by removing the descriptions of useless actions from the definition of a planning problem we may be able to reduce the size of the problem and make the planning task easier. Third, we will see later that, besides being more economical, the plans that do not contain irrelevant actions have also another advantage: they remain valid after removing "irrelevant assumptions" about the initial state.

In our analysis of the idea of a useless action, we assume that the given action domain is described in language $\mathcal{C}+$ [Giunchiglia *et al.*, 2004]—one of the "new generation" action description languages that are more expressive in some ways[2] than the "classical" languages STRIPS and ADL. The use of such an expressive representation language makes the investigation of irrelevant actions a more challenging problem.

2 Example

Our approach is illustrated here using a simplified version of the hat example that involves three missionaries and three hats, but no cannibals. Consider first the $\mathcal{C}+$ description of the "three missionaries" domain shown in Fig. 1. The main part of this description consists of the three causal laws describing the effects and preconditions of action $Cross(m)$ ("missionary m crosses the river"). First, executing this action causes both m and the boat to be at the bank l different from the bank where m was before. Second, this action cannot be executed unless m and the boat are on the same bank. Third, the boat only holds two.

[1] This has been observed in some of the experiments mentioned in Sect. 6 below.

[2] For instance, $\mathcal{C}+$ allows us to describe effects of actions indirectly. CCALC (see `http://www.cs.utexas.edu/users/tag/ccalc/`) is a planner for action domains represented in $\mathcal{C}+$ that is based on ideas of satisfiability planning.

Notation: i, j are integers such that $1 \leq i < j \leq 3$.

Simple fluent constants: $Owner(H_1),\ Owner(H_2),\ Owner(H_3)$

Domains: $\{M_1, M_2, M_3\}$

Boolean action constants: $Exchange(H_1, H_2),\ Exchange(H_1, H_3),\ Exchange(H_2, H_3)$

Causal laws:

constraint $Owner(H_i) \neq Owner(H_j)$

$Exchange(H_i, H_j)$ **causes** $Owner(H_i) = m' \wedge Owner(H_j) = m$
if $Owner(H_i) = m \wedge Owner(H_j) = m'$

nonexecutable $Exchange(H_i, H_j)$
if $Owner(H_i) = m \wedge Owner(H_j) = m' \wedge Loc(m) \neq Loc(m')$

nonexecutable $Exchange(H_i, H_j)$
if $Owner(H_i) = m \wedge Owner(H_j) = m' \wedge \neg(Cross(m) \equiv Cross(m'))$

Fig. 2. Additional actions: exchanging hats. When these postulates are added to Fig. 1, the schemas in the last two lines of Fig. 1 are understood to apply to the new fluent and action constants as well

The planning problem P_1, in which all missionaries and the boat are initially on $Bank_1$ and their goal is to get to $Bank_2$, can be solved in 3 steps:

1. $\{Cross(M_1),\ Cross(M_2)\}$
2. $\{Cross(M_1)\}$
3. $\{Cross(M_1),\ Cross(M_3)\}$

We call this solution Plan A.

Consider now the elaboration of the three missionaries domain obtained by adding hats (Fig. 2). According to the first of the four additional postulates, a person does not wear two hats simultaneously. The second postulate describes the effect of the action $Exchange(H_i, H_j)$ (exchanging the hats H_i and H_j). Finally, this action is assumed to be nonexecutable if the owners of the hats are on different banks of the river, and it cannot be executed concurrently with crossing the river if one of the owners is crossing and the other is not.

In the extended domain, consider the planning problem P_2 obtained by adding to P_1 the additional initial conditions specifying the owners of all hats: $Owner(H_i) = M_i$ $(i = 1, 2, 3)$. Plan A solves P_2 as well. Besides, P_2 has solutions involving the action of exchanging hats—for instance, the following Plan B, in which M_1 and M_2 exchange their hats while crossing the river:

1. $\{Cross(M_1),\ Cross(M_2),\ Exchange(H_1, H_2)\}$
2. $\{Cross(M_1)\}$
3. $\{Cross(M_1),\ Cross(M_3)\}$

But this plan can be "improved" by dropping $Exchange(H_1, H_2)$. Generally, if a sequence of events $e_1, \ldots, e_n$ is a solution to the problem P_2, then the sequence $e_1 \setminus E, \ldots, e_n \setminus E$, where E stands for $\{Exchange(H_i, H_j) : 1 \leq i < j \leq 3\}$, is a solution too. In this sense, the action of exchanging hats is irrelevant.

Besides being more economical, Plan A is also more attractive than Plan B for another reason, which is related to "conformant planning" [Smith and Weld, 1998]—generating plans without branching when the initial conditions are incomplete (or actions are nondeterministic). Imagine, for instance, that the formulas $Owner(H_i) = M_i$ are dropped from the list of initial conditions, so that the given planning problem has 6 possible initial states. We want to find a plan that guarantees achieving the goal in each of these states. Plan A has this property. Plan B does not: in some initial states, the redundant action $Exchange(H_1, H_2)$ is not executable concurrently with $Cross(M_1)$ and $Cross(M_2)$, because the owner of one of the hats H_1, H_2 may be crossing the river while the other is not.

In the following sections we make the idea of isolated parts of an action description precise under the assumption that the action description is written in $\mathcal{C}+$. Then we show that our two observations about Plan B—that it can be made more economical and that it can be turned into a solution to a conformant planning problem—are instances of a general theorem about isolated sets.[3]

3 Action Descriptions, Transition Systems, and Plans

The definitions of the syntax and semantics of language $\mathcal{C}+$ can be found in [Giunchiglia *et al.*, 2004, Sect. 4]. Recall that the *signature* of an action description in this language consists of symbols of two kinds—*fluent constants* and *action constants*. According to the syntax of $\mathcal{C}+$, an *action description* is a set of "causal laws" of three kinds—static laws, action dynamic laws, and fluent dynamic laws. Causal laws of the first two kinds have the form

$$\textbf{caused } F \textbf{ if } G,$$

and fluent dynamic laws have the form

$$\textbf{caused } F \textbf{ if } G \textbf{ after } H$$

(F, G, H are formulas satisfying certain syntactic conditions). In Figs. 1 and 2 we use abbreviations for causal laws of special types. For instance,

$$\textbf{nonexecutable } a \textbf{ if } F$$

stands for the fluent dynamic law

$$\textbf{caused } \bot \textbf{ if } \top \textbf{ after } a \wedge F.$$

See [Giunchiglia *et al.*, 2004, Appendix B] for a complete list of abbreviations.

According to the semantics of $\mathcal{C}+$, every action description denotes a "transition system"—a directed graph whose vertices correspond to states, and whose

[3] This paper does not address the problem of conformant planning in full generality. The example above is special in the sense that its initial conditions uniquely determine the values of all "relevant" fluent constants.

edges correspond to events leading from one state to another. Every state is a function that maps fluent constants to their values. Every event is a function that maps action constants to their values. In the simple case when all action constants are Boolean, as in the example in Sec. 2, an event e is understood as the concurrent execution of all actions a such that $e(a) = \mathbf{t}$, and we identify e with the set of all such actions.

Consider a transition system TS. A *plan* is a finite sequence of events labeling the edges of TS. A *history* is an arbitrary path $\langle s_0, e_0, s_1, \ldots, e_{n-1}, s_n \rangle$ in TS ($s_0, \ldots, s_n$ are states, and $e_0, \ldots, e_{n-1}$ are events); n is the *length* of the history. Histories of length 1 (that is, edges of TS) are called *transitions*.

The following definitions are based on [McCain and Turner, 1998]. We define recursively when a plan is *executable* in a given state:

(i) The empty plan $\langle\,\rangle$ is executable in any state;
(ii) a non-empty plan $\langle e_0, \ldots, e_n \rangle$ is executable in a state s_0 if
  - $\langle e_0, \ldots, e_{n-1} \rangle$ is executable in s_0, and
  - for any states $s_1, \ldots, s_n$ such that $\langle s_0, e_0, s_1, \ldots, e_{n-1}, s_n \rangle$ is a history, there exists a state s_{n+1} such that $\langle s_n, e_n, s_{n+1} \rangle$ is a transition.

Let G be a formula that does not contain action constants. We say that a plan $\langle e_0, \ldots, e_{n-1} \rangle$ is *sufficient* for achieving the goal G in a state s_0 if, for any states $s_1, \ldots, s_n$ such that $\langle s_0, e_0, s_1, \ldots, e_{n-1}, s_n \rangle$ is a history, s_n satisfies G.

A plan is *valid* for achieving G in a state s_0 if it is executable in s_0 and sufficient for achieving G in s_0.

Consider, for instance, the "missionaries and hats" action description MH, shown in Figs. 1 and 2. Let the goal G be

$$Loc(M_1) = Bank_2 \wedge Loc(M_2) = Bank_2 \wedge Loc(M_3) = Bank_2. \tag{1}$$

Both Plan A and Plan B defined in Sect. 2 are valid for achieving the goal G in the state satisfying the conditions

$$Loc(M_i) = Loc(Boat) = Bank_1, \;\; Owner(H_i) = M_i \;\; (i = 1, 2, 3). \tag{2}$$

Plan A is also valid for achieving G in the state

$$\begin{array}{l} Loc(M_i) = Loc(Boat) = Bank_1 \;\; (i = 1, 2, 3), \\ Owner(H_1) = M_1, Owner(H_2) = M_3, Owner(H_3) = M_2, \end{array} \tag{3}$$

but Plan B is not: in this state it is not executable.

4 Isolated Sets

Consider an action description D in the language $\mathcal{C}+$ with the set σ^{fl} of fluent constants and the set σ^{act} of action constants. By σ^{all} we denote $\sigma^{fl} \cup \sigma^{act}$.

A subset σ of σ^{all} is *isolated* with respect to D if, for every causal law $L \in D$,

(i) all constants occurring in L belong to σ, or

(ii) L does not contain constants from σ, or
(iii) L is an action dynamic law of the form

$$\textbf{caused } F \textbf{ if } H \wedge c \wedge H' \tag{4}$$

or a fluent dynamic law of the form

$$\textbf{caused } F \textbf{ if } G \textbf{ after } H \wedge c \wedge H', \tag{5}$$

where c is a Boolean action constant that does not belong to σ, and F and G are formulas that do not contain constants from σ.

In other words, σ is isolated if constants from σ and constants from $\sigma^{all} \setminus \sigma$ can occur together in a causal law from D only when that causal law has the form (4) or (5), with constants from σ occurring in the H and H' parts only.

For instance, in the case of action description MH,

$$\sigma^{fl} = \{Loc(M_i) : 1 \leq i \leq 3\} \cup \{Loc(Boat)\} \cup \{Owner(H_i) : 1 \leq i \leq 3\},$$
$$\sigma^{act} = \{Cross(M_i) : 1 \leq i \leq 3\} \cup \{Exchange(H_i, H_j) : 1 \leq i < j \leq 3\}.$$

Define

$$\sigma = \{Loc(M_i) : 1 \leq i \leq 3\} \cup \{Loc(Boat)\} \cup \{Cross(M_i) : 1 \leq i \leq 3\}. \tag{6}$$

This set is isolated with respect to MH. Indeed, the only causal laws in MH that contain both constants from σ and constants from $\sigma^{all} \setminus \sigma$ are the last two in Fig. 2, which stand for causal laws of form (5) with $Exchange(H_i, H_j)$ as c.

5 Main Theorem

Let σ be a subset of σ^{all}. States s and s' are *σ-equivalent* to each other if the value of every fluent constant from σ in s is the same as its value in s'. For instance, if σ is defined by (6) then state (2) is σ-equivalent to state (3).

In the theorem below, D is an action description whose action constants are Boolean. The statement of the theorem refers to the following condition on D:

(*) $\langle\emptyset\rangle$ is a plan executable in every state.

This condition expresses that for every state s there exists a state s_1 such that $\langle s, \emptyset, s_1\rangle$ is a transition. For many action descriptions, including MH, it holds with $s_1 = s$.

Main Theorem *Assume that D satisfies (*). Let σ be a subset of σ^{all} that is isolated with respect to D, and let G be a formula of the signature $\sigma^{fl} \cap \sigma$. If $\langle e_0, \ldots, e_{n-1}\rangle$ is a plan valid for achieving the goal G in a state s_0 then the sequence $\langle e_0 \cap \sigma, \ldots, e_{n-1} \cap \sigma\rangle$ is a plan valid for achieving G in any state that is σ-equivalent to s_0.*

The theorem tells us that if all fluent constants occurring in the goal of a planning problem belong to an isolated set σ then dropping the action constants that do not belong to σ from a valid plan will result in a valid plan also.

For instance, the transformation of Plan B into Plan A in Sect. 2 by dropping $\mathit{Exchange}(H_1, H_2)$ corresponds to σ defined by (6), and to formula (1) as G. Furthermore, the "more economical" plan is valid not only in state s_0, but also in any state that is σ-equivalent to s_0. In the example from Sect. 2, from the fact that Plan B is valid in state (2) and all constants occurring in the goal (1) belong to (6) we can conclude that Plan A is valid for achieving this goal not only in state (2), but also in state (3).

The proof of the main theorem can be found in the complete version of the paper (`http://www.cs.utexas.edu/users/vl/papers/iaf.ps`).

Without condition (*), the statement of the theorem would be incorrect.

6 Simplifying Action Descriptions

From the main theorem we know that, under assumption (*), a solvable planning problem remains solvable if we limit attention to the plans that do not contain irrelevant actions. This limitation can be expressed by declaring the irrelevant actions nonexecutable. Indeed, the effect of adding to D the causal law

$$\textbf{nonexecutable } a \tag{7}$$

is to remove from the corresponding transition system the edges whose labels contain a. Furthermore, in the presence of (7), removing causal laws of the forms

$$\textbf{nonexecutable } a \textbf{ if } F$$

and

$$a \textbf{ causes } F \textbf{ if } G$$

will have no effect on the transition system. Such transformations can be used to simplify the statement of a planning problem by removing postulates.

Consider, for instance, our missionaries and hats example—the planning problem P_2 for action description MH, with initial conditions (2) and goal (1). If we are only interested in the solutions that do not contain the irrelevant actions $\mathit{Exchange}(H_i, H_j)$ then we can remove the edges whose labels contain these actions by adding to MH the causal laws of form (7) with $\mathit{Exchange}(H_i, H_j)$ as a. After that, the action description can be simplified by dropping the last three causal laws from Fig. 2.

These observations suggest that the main theorem of this paper can be sometimes used to replace planning problems with isolated parts by equivalent, smaller problems that may be solved faster. This is confirmed by our experiments on the use of CCalc for solving the missionaries and hats problem (see the full paper for details).

7 Conclusion

In this paper we defined the concept of an isolated set of constants for action descriptions in language $\mathcal{C}+$. By identifying an isolated set we can detect constants that are irrelevant for a given planning problem. After dropping irrelevant actions, a valid plan remains valid for the same initial state, and it may also become valid for other initial states. By declaring the irrelevant actions nonexecutable we can sometimes make the planning problem easier.

Partitioning theories was studied both in the context of classical logic [Amir, 2001] and for nonmonotonic theories, such as logic programs under the answer set semantics [Lifschitz and Turner, 1994]. The splitting set theorem from the last paper, in combination with Proposition 1 from [Lifschitz and Turner, 1999], can be actually used to prove a special case of our main theorem.

We expect that the main theorem of this paper can be used to extend the computational possibilities of CCALC. To this end, we need to be able to find the minimal sets containing all constants occurring in the goal of a given planning problem. It turns out that such a set is always unique, and there is a simple algorithm for computing it.

Acknowledgements

We would like to thank Jonathan Campbell, Selim Erdoğan, Paolo Ferraris, Michael Gelfond, Joohyung Lee, Yuliya Lierler, Hudson Turner and anonymous referees for comments. This work was partially supported by the Texas Higher Education Coordinating Board under Grant 003658-0322-2001.

References

[Amir, 2001] Eyal Amir. *Dividing and Conquering Logic.* PhD thesis, Stanford University, 2001.

[Giunchiglia *et al.*, 2004] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1–2):49–104, 2004.

[Kautz and Selman, 1992] Henry Kautz and Bart Selman. Planning as satisfiability. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*, pages 359–363, 1992.

[Lifschitz and Turner, 1994] Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In Pascal Van Hentenryck, editor, *Proceedings of International Conference on Logic Programming (ICLP)*, pages 23–37, 1994.

[Lifschitz and Turner, 1999] Vladimir Lifschitz and Hudson Turner. Representing transition systems by logic programs. In *Logic Programming and Non-monotonic Reasoning: Proc. Fifth Int'l Conf. (Lecture Notes in Artificial Intelligence 1730)*, pages 92–106, 1999.

[McCain and Turner, 1998] Norman McCain and Hudson Turner. Satisfiability planning with causal theories. In Anthony Cohn, Lenhart Schubert, and Stuart Shapiro, editors, *Proc. Sixth Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 212–223, 1998.

[McCarthy, 1998] John McCarthy. Elaboration tolerance.[4] In progress, 1998.

[Smith and Weld, 1998] David E. Smith and Daniel S. Weld. Conformant graphplan. In *Proc. AAAI-98*, pages 889–896, 1998.

[4] `http://www-formal.stanford.edu/jmc/elaboration.html` .