

A Sound Linear Algorithm for Pre-processing planning problems with Language Axioms

Max Garagnani *

Abstract

Given a planning domain, the choice of the language for the state definition, goals and operator specification is, often, not simple: if a language is sophisticated enough to allow an accurate and natural description of these elements, it is likely to contain expressions which are related by *logical formulae*, determined by the specific characteristics of that domain.

In this paper, a clear distinction between the state definition (or ‘*inner*’) language Li and its (‘*outer*’) extension $Lo \supset Li$ used to specify goals and preconditions is adopted. According to such distinction, the logical relations (or ‘Language Axioms’, LAs) holding between the expressions of Lo are explicitly represented and considered as elements of the planning problem.

The problem of planning with LAs using a given planner which does not support them is hence tackled: following a *pre-processing* approach, a linear algorithm for the automatic transformation of planning problems containing LAs with atomic premisses into *equivalent* ones which do not is described, and its correctness formally proved.

1 Introduction

Given a planning domain, the choice of the expressions which will constitute the language for the state description is, often, not simple. On the one hand, the language should be expressive enough to allow an easy and natural ‘translation’ from the real situation into the planning domain. On the other hand, the more expressive the language, the higher the number of *interactions* between different expressions, and the more difficult to deal with them during the planning process.

For example, consider a block-world domain in which the five following predicates constitute the domain-definition language Lo :

$$Lo = \{ \text{on_top}(x, y), \text{clear}(x), \text{on_table}(x), \text{above}(x, y), \text{under}(x, y) \}$$

* Department of Computing, Faculty of Mathematics and Computing, The Open University, Walton Hall, Milton Keynes, MK7 6AA, U.K. E-mail: m.garagnani@open.ac.uk

where the predicate ‘above(x, y)’ (and ‘under(x, y)’) indicates that the block ‘ x ’ is on the same pile of ‘ y ’, but in a higher (lower) position. It is easy to identify various logical relations holding between the expressions of Lo :

$$\begin{aligned}
\forall x, \forall y \quad & (\text{on_top}(x, y) \rightarrow \text{above}(x, y)) \\
\forall x, \forall y \quad & (\text{on_top}(x, y) \wedge \text{above}(y, z) \rightarrow \text{above}(x, z)) \\
\forall x, \forall y \quad & (\text{above}(x, y) \rightarrow \neg \text{on_table}(x) \wedge \neg \text{clear}(y)) \\
\forall x, \forall y \quad & (\text{clear}(x) \rightarrow \neg \text{above}(y, x)) \\
\forall x, \forall y \quad & (\text{above}(x, y) \rightarrow \text{under}(y, x)) \\
& \dots \quad \dots
\end{aligned}$$

The presence of such language (or domain) axioms makes the planning problem more complex. However, the adoption of an expressive language allows the direct specification of *preconditions* and *goals* which otherwise would require a cumbersome representation. In fact, for example, suppose the goal of a four-blocks problem to consist simply of achieving above(A,B). If neither the ‘above’ nor the ‘under’ relations were included in Lo , the given goal would have to be translated into the following complex expression:

$$\begin{aligned}
\text{above}(A, B) \equiv & \text{on_top}(A, B) \vee (\text{on_top}(A, C) \wedge \text{on_top}(C, B)) \\
& \vee (\text{on_top}(A, D) \wedge \text{on_top}(D, B)) \\
& \vee (\text{on_top}(A, C) \wedge \text{on_top}(C, D) \wedge \text{on_top}(D, B)) \\
& \vee (\text{on_top}(A, D) \wedge \text{on_top}(D, C) \wedge \text{on_top}(C, B))
\end{aligned}$$

Notice that if the number of blocks increases, the number of terms of the previous disjunction grows exponentially.

Empirical evidence suggests that, given a domain language Lo (from now on identified as ‘outer’ language) expressive enough to contain language axioms (or ‘LAs’), it is always possible to identify an ‘inner’ language $Li \subset Lo$ such that Li alone is sufficient to describe unequivocally any possible state of that domain. Moreover, for the specific Li , there exists a set of LAs such that any ‘outer’ state — described using Lo — can be obtained from an *equivalent* ‘inner’ state (subset of Li) through the application of the axioms.

Example 1.1 Reconsider the outer language Lo specified before and the three axioms

$$\begin{aligned}
\text{on_top}(x, y) & \rightarrow \text{above}(x, y) & (1) \\
\text{on_top}(x, y) \wedge \text{above}(y, z) & \rightarrow \text{above}(x, z) & (2) \\
\text{above}(x, y) & \rightarrow \text{under}(y, x) & (3)
\end{aligned}$$

The inner language

$$Li = \{ \text{on_top}(x, y), \text{clear}(x), \text{on_table}(x) \} \subset Lo$$

can be adopted as state definition language for the domain, since the remaining propositions in $Lo \setminus Li$ can be directly derived from those in Li . In fact, any state So defined using Lo can be reduced to an *equivalent* state Si (defined in Li) by removing any ‘above()’ or ‘under()’ relation from it. Through the application of the axioms (1)–(3), then, the original state So can be ‘reconstructed’ again.

By way of example, given the inner state

$$Si = \{ \text{on_top}(A,B), \text{on_top}(B,C), \text{on_table}(C), \text{clear}(A) \}$$

we would obtain:

$$So = Si \cup \{ \text{above}(A,B), \text{above}(A,C), \text{under}(B,A), \text{under}(C,A) \}$$

Notice that not only is the set of axioms (1)–(3) sufficient to reconstruct any possible outer state $So \subseteq Lo$, but it also represents a *minimal* set: indeed, each of the axioms is *necessary* for the correct completion of the process of deduction.

It is not difficult to see that the set $(Lo \setminus Li)$ corresponds to the set of expressions which appear as *consequences* of the LAs. Moreover, since the state is defined using Li only, the *add* and *delete* lists of the operators should contain only inner language propositions, whilst *preconditions* and *goals* will be allowed to contain any outer-language expression.

By relying on the previous considerations, the rest of the paper illustrates an algorithm for the pre-processing of any planning problem which contains LAs in the form $p \vdash c$, where ‘ p ’ and ‘ c ’ are atomic *wff*’s representing premiss and consequence of an axiom. The output of such a process will consist of a problem *equivalent* to the original but containing no LAs. It should be pointed out that, for reasons of space and clarity, this work will not examine the problem of pre-processing axioms with multiple premisses, such as $p_1 \wedge p_2 \wedge \dots \wedge p_n \vdash c$. Gazen and Knoblock [5] have proposed a possible solution to this problem; however, their approach presents various drawbacks, already analysed in [3]. A possible improvement of their method is described in [4].

2 Definitions

In what follows, the terminology adopted is in accordance with the definitions presented in [2] and [3]. More precisely, Li and Lo are propositional (inner and outer) languages such that $Li \subset Lo$; given a set R of LAs for Lo , the *closure* (or ‘outer extension’) $C_R(S)$ of $S \subseteq Li$ is defined as

$$C_R(S) = \{ \omega \in Lo \mid S \vdash_R \omega \} \tag{4}$$

where $S \vdash_R \omega$ means that ω can be derived from S using the axioms in R ¹. From this definition, it follows immediately that

$$\forall S \subseteq Li, S \subseteq \mathbf{C}_R(S) \quad (5)$$

A set R of LAs is ‘*acyclic*’ iff $\forall r \in R$, the consequence ‘ c ’ of the axiom r never appears in the premisses of the axioms in R .

A ‘*LA-planning problem*’ is a classical planning problem (initial state, goals and operator schemes) augmented with a set R of language axioms. More formally, a LA-planning problem consists of a tuple (I, Op, G, LA^*) such that $I \subseteq Li$ is the initial state, Op is a set of operator schemes in the form (P, A, D) (with $P \subseteq Lo$, $A \subseteq Li$, $D \subseteq Li$), $G \subseteq Lo$ is the set of goals, and $LA^*(\cdot)$ is the function which calculates the closure $\mathbf{C}_R(\cdot)$ for the given set R of LAs.

Given a LA-planning problem $\mathcal{P} = (I, Op, G, LA^*)$, a *plan* for \mathcal{P} consists of a sequence of instances of operators $\langle O_0, O_1, \dots, O_n \rangle$ (with $O_i \in Op$) such that, if S_i is the state before the application of the operator O_i and F is the final state, the following two conditions are true:

1. for each step O_i in the plan, if P is the precondition set of O_i , then $\forall p \in P, p \in LA^*(S_i)$;
2. $\forall g \in G, g \in LA^*(F)$.

Notice that according to the given definition of LA-planning problem, a ‘standard’ planning problem can be considered as a particular instance of LA-planning problem having $LA^* = Id^2$ and $Lo = Li$.

Finally, in order for two planning problems $\mathcal{P}, \mathcal{P}'$ to be considered *equivalent*, there must be a *bijective* function which associates every plan solution of \mathcal{P} with one (and only one) solution of \mathcal{P}' , and vice versa.

3 Transforming a LA-planning problem

This section describes an algorithm for the transformation of a LA-planning problem $\mathcal{P} = (I, Op, G, LA^*)$ into an equivalent problem $\mathcal{P}' = (I', Op', G', LA_1^*)$ having $LA_1^* = Id$. If the function LA_1^* calculates the identity, its presence in the problem is irrelevant, and \mathcal{P}' can be solved normally, as a standard planning problem (I', Op', G') .

The transformation is realized by rendering fully explicit the LA-planning problem \mathcal{P} , that is, by completing *every state* — and every operator’s effects — with all the expressions of Lo which would have been ‘visible’ through the application of the axioms. Hence, \mathcal{P}' will be entirely defined in an outer-language

¹This definition of closure is equivalent to that of *deductive closure* given by Konolige in [6].

² Id represents the ‘identity’ function, i.e. $Id(S) = S, \forall S \subseteq Li$.

‘world’, and will not require the use of any closure function or LAs for the construction of the plan solution.

The procedure for the pre-processing of \mathcal{P} with axiom set R relies upon two major hypotheses:

1. it assumes R acyclic³;
2. it assumes R to contain only LAs in the form $p \vdash c$, with $c \in (Lo \setminus Li)$.

Reported below are the steps necessary to build the elements I' , Op' and G' of the new problem \mathcal{P}' . Notice that the expression $Prem(x)$, used in the algorithm, should be read as ‘premisses of x ’, and is defined as the set $Prem(x) = \{p_1, \dots, p_k\}$ containing the premisses of the LAs in R which have consequence ‘ x ’ and premiss $p_i \notin D$ ⁴.

ALGORITHM A1

- $I' = \mathbf{C}_R(I)$;
- $Op' = \Gamma(Op)$,
 where $\forall O' \in Op'$, $O' = \gamma(O)$, and $\gamma(P, A, D)$ is an operator with *conditional effects* $O' = [(P', A', D'), e_1, \dots, e_n]$, such that
 - $P' = P$;
 - $A' = \mathbf{C}_R(A)$;
 - $D' = \emptyset$;
 - the list of conditional effects $[e_1, \dots, e_n]$ is built in two separate steps:
 1. $\forall d \in D$, if $d \notin A'$, append to the list the conditional deletion

$$e_i = (P_i, A_i, D_i) = (\neg Prem(d), \emptyset, \{d\})$$
 2. $\forall e_i$ appended in the previous step such that $P_i = \emptyset$, consider the proposition deleted d : $\forall c \in (\mathbf{C}_R(\{d\}) \setminus \{d\})$, if $c \notin A'$ then append to the list the conditional deletion

$$e_j = (P_j, A_j, D_j) = (\neg Prem(c), \emptyset, \{c\})$$
- $G' = G$.

Let us now explain, step by step, the meaning of the previous transformation.

³It will be shown later on that this condition can actually be replaced by a weaker requirement.

⁴The ‘negation’ of this expression, written as $\neg Prem(x)$, represents the set of negated propositions $\{\neg p_1, \neg p_2, \dots, \neg p_k\}$, which corresponds, when interpreted as a conjunction, to the boolean expression $(\neg p_1 \wedge \neg p_2 \wedge \dots \wedge \neg p_k)$. Hence, if $Prem(x) = \emptyset$, then $\neg Prem(x) = True$.

3.1 States and Language of \mathcal{P}'

Intuitively, the transformation A1 integrates into the planning problem itself the *effects* of the function $\mathbf{C}_R()$. As a result, in the new problem \mathcal{P}' each state S_i will explicitly contain all the propositions which would have been added by calculating $\mathbf{C}_R(S_i)$. It follows that the new ‘extended’ initial state I' should be defined as $I' = \mathbf{C}_R(I)$.

Moreover, because of this extension, the states of \mathcal{P}' can contain expressions which, in the original problem \mathcal{P} , did not belong to Li . Hence, the new problem will be defined using an extended inner language $Li' \subseteq Lo$, obtained as

$$Li' = \mathbf{C}_R(Li)$$

Notice that the new outer language Lo' remains identical to Lo .

3.2 Operators of \mathcal{P}'

Each operator $O \in Op$ is transformed into a new operator $O' = \gamma(O)$ which may contain conditional effects. The presence of conditional effects is due to integration of the consequences of the LAs into the effects of the operator.

Let $O = (P, A, D)$ be the operator which is being considered. The new operator O' will consist of a list of triples $[(P', A', D'), e_1, e_2, \dots, e_n]$, where the head (P', A', D') represents the main effect of the operator, and $e_i = (P_i, A_i, D_i)$ is a conditional effect.

The new precondition list P' will simply be equivalent to the original P , which was already expressed using the outer language Lo .

Consider now the add list A' . The new operator O' must add all the propositions which would have been added by O . Hence, $A \subseteq A'$. Moreover, O' should also add all the new propositions that the closure $\mathbf{C}_R(S \cup A)$ would have contained after the addition of A to the ‘extended’ state $\mathbf{C}_R(S)$, that is, the set $\mathbf{C}_R(S \cup A) \setminus \mathbf{C}_R(S)$. Because of the atomic premiss of the axioms in R , it is possible to prove that $\forall A, B \subseteq Li, \mathbf{C}_R(A \cup B) = \mathbf{C}_R(A) \cup \mathbf{C}_R(B)$ ⁵. Hence, $\mathbf{C}_R(S \cup A) \setminus \mathbf{C}_R(S) = \mathbf{C}_R(A)$. Since $\mathbf{C}_R(A)$ contains also A , we can define $A' = \mathbf{C}_R(A)$.

The new delete list D' will be (initially) left empty. However, at the end of steps 1 and 2 of A1, each new conditional deletion e_i appended to the operator such that $P_i = \emptyset$ can be transferred into the main delete list D' .

Let us now explain the reasons for the introduction of the list of conditional deletions in the operators. Consider the original operator O , deleting all the propositions in D . Suppose that the set R contains an axiom $p \vdash d$ such that $d \in D$. If the proposition p is present in the state after the application of O ,

⁵In fact, since the premiss of any axiom in R contains only one atomic proposition, every consequence of the set A will also be consequences of $A \cup B$; *idem* for the consequences of B . Hence, $\mathbf{C}_R(A) \cup \mathbf{C}_R(B) \subseteq \mathbf{C}_R(A \cup B)$. The vice versa holds because every consequence ‘ c ’ of $A \cup B$ is generated by a premiss $p \in A \cup B$ which belongs to A or to B (or to both).

the closure \mathbf{C}_R of this state will contain, again, the proposition d , as the axiom $p \vdash d$ will re-generate d from p .

Therefore, the elements of D will have to be deleted by O' under *specific conditions*, that is, that each considered proposition d be not the *consequence* of any rule $p \vdash d$ such that p is present in the current state (or will be present in the state after the application of the operator O). Thus, each proposition $d \in D$ (which *does not belong to A'*) will produce the addition (step 1 of A1) of a *conditional* deletion

$$e_i = (P_i, A_i, D_i) = (\neg Prem(d), \emptyset, \{d\})$$

to the list (initially empty) of conditional effects $[e_1, \dots, e_n]$ in O' ⁶. In other words, d will be deleted by O' iff it would have not appeared in the closure $\mathbf{C}_R(S \cup A \setminus D)$, where S is the state to which the operator O is applied.

Consider now the list $L_D = [e_1, \dots, e_n]$ of conditional effects added by step 1, with $e_i = (\neg Prem(d), \emptyset, \{d\})$. Suppose a proposition d constitutes the premiss of one of the inference rules of R , e.g. $d \vdash c$. Because of the *acyclicity* of R , this can happen only when d is not a consequence of any axiom in R (and, thus, $Prem(d) = \emptyset$). Since the condition $\neg Prem(d)$ is verified, and d is deleted, should c be deleted too?

The answer is ‘yes’ *iff* its presence in the state was caused solely by the presence of d . Indeed, there might be another proposition q in the current state (not deleted by the operator) such that $q \vdash c$, in which case c should not be deleted. Notice that c could not have been present in the initial state $I \subseteq Li$, as, for hypothesis, $c \in Lo \setminus Li$, and thus $c \notin Li$; neither could it have been added by the ‘main’ effect A of one of the previous operators, for every add list A must be a subset of Li , too. Therefore, if c is in the state, it must be a consequence of one (or many) of the axioms.

Hence, given d such that $(\emptyset, \emptyset, \{d\}) = e_i = (P_i, A_i, D_i) \in L_D$ and $(r) d \vdash c$, where r is one of the LAs in R , if $c \notin A'$, the following conditional deletion should be appended to the list of effects of O' (step 2 in A1):

$$e_j = (P_j, A_j, D_j) = (P_i \wedge \neg Prem(c), \emptyset, \{c\}) = (\neg Prem(c), \emptyset, \{c\}) \quad (6)$$

This process must be repeated for each $e_i = (\emptyset, \emptyset, \{d\})$ which appears in the list L_D . Notice, though, that because of the hypothesis of *acyclicity* of R , it is not necessary to *iterate* the process and reconsider the deletions possibly produced by this second step. This follows from the fact that all the propositions c conditionally deleted by these effects are consequences of rules, and cannot appear as premisses of other rules of R . Hence, their (possible) deletion can not produce any further propagatory effect.

⁶Notice that the use of the expression $\neg Prem(d)$ as precondition list requires the negated premisses $\neg p_1, \dots, \neg p_k$ to be expressed each by a *single wff* of Lo .

Example 3.1 Consider the languages Lo and Li of Example 1.1 and the set R containing the two following axioms:

$$\begin{aligned} \text{on_top}(x, y) &\vdash \text{above}(x, y) \\ \text{on_top}(x, y) &\vdash \text{under}(y, x) \end{aligned}$$

The operator ‘ $\text{put_down}(x, y)$ ’, consisting of

$$\begin{aligned} P &= \{\text{on_top}(x, y), \text{clear}(x)\} \\ A &= \{\text{on_table}(x), \text{clear}(y)\} \\ D &= \{\text{on_top}(x, y)\} \end{aligned}$$

will undergo the following transformation:

$$\begin{aligned} P' &= P \\ A' &= \mathbf{C}_R(A) = A \\ D' &= \emptyset \end{aligned}$$

$$\begin{aligned} \text{Step1 : } e_1 &= (\emptyset, \emptyset, \{\text{on_top}(x, y)\}) \\ &(\text{since } \neg \text{Prem}(\text{on_top}(x, y)) = \emptyset) \end{aligned}$$

$$\begin{aligned} \text{Step2 : } \mathbf{C}_R(\{\text{on_top}(x, y)\}) \setminus \{\text{on_top}(x, y)\} &= \{\text{above}(x, y), \text{under}(y, x)\} \\ e_2 &= (\emptyset, \emptyset, \{\text{above}(x, y)\}) \\ e_3 &= (\emptyset, \emptyset, \{\text{under}(y, x)\}) \end{aligned}$$

Notice that $\text{Prem}(c)$ for $c \in \{\text{above}(x, y), \text{under}(y, x)\}$ is empty as $\text{on_top}(x, y) \in D$. In summary, if we join the conditional deletions having empty precondition with the main delete list D' we obtain:

$$\begin{aligned} P' &= P \\ A' &= A \\ D' &= D \cup \{\text{above}(x, y), \text{under}(y, x)\} \end{aligned}$$

4 Formal Analysis

This section contains the formal proof of the soundness of the algorithm A1. The proof is based on a property concerning the transformations \mathbf{C}_R and Γ . Consider, in the original planning problem \mathcal{P} , the change produced by an operator O applied to the initial state I

$$I \xrightarrow{O} S_1$$

indicating that the application of O to I produces the new state S_1 .

The new problem \mathcal{P}' starts with an 'extended' initial state $I' = \mathbf{C}_R(I)$, and adopts a new set of (extended) operators $Op' = \Gamma(Op)$, where Γ is realized by transforming each operator $O = (P, A, D) \in Op$ into a new conditional-effect operator $O' = \gamma(O)$ (notice that γ is $1-1$). Because of the way in which the transformation of the operators has been defined, the corresponding change in the new problem-world

$$\mathbf{C}_R(I) \xrightarrow{\gamma(O)} S'_1$$

produces a state S'_1 which is *exactly* $\mathbf{C}_R(S_1)$. In other words, every step

$$S_t \xrightarrow{O} S_{t+1}$$

of a plan in the original problem can be mapped to a *corresponding* step of the new problem plan

$$\mathbf{C}_R(S_t) \xrightarrow{\gamma(O)} \mathbf{C}_R(S_{t+1})$$

and vice versa. More formally, given two states S_t and $\mathbf{C}_R(S_t)$, and two operators $O \in Op$ and $O' = \gamma(O) \in \Gamma(Op)$, the application of O to S_t and of O' to $\mathbf{C}_R(S_t)$

$$\begin{array}{ccc} S_t & \xrightarrow{O} & S_{t+1} \\ \downarrow & & \\ \mathbf{C}_R(S_t) & \xrightarrow{\gamma(O)} & S'_{t+1} \end{array} \quad (7)$$

produces two states S_{t+1} and S'_{t+1} such that $S'_{t+1} = \mathbf{C}_R(S_{t+1})$.

Because of this property, since the plan of the new problem \mathcal{P}' starts from an initial state $I' = \mathbf{C}_R(I)$, then every following state $\mathbf{C}_R(S_t)$ of the plan in \mathcal{P}' (including the final state) will correspond to a state S_t of a plan in \mathcal{P} . Therefore, the two LA-planning problems \mathcal{P} and \mathcal{P}' can be shown to be equivalent. The formal proof of this equivalence is the object of the following theorem:

Theorem 4.1 (A1-equivalence) *Let $\mathcal{P} = (I, Op, G, LA^*)$ be a LA-planning problem, with $LA^* = f \circ f_i$, where $f_i = \mathbf{C}_R$, and the set R of inference rules on Lo is acyclic and contains only rules in the form $p \vdash c$, with $c \in Lo \setminus Li$. Then, the new LA-planning problem $\mathcal{P}' = (\mathbf{C}_R(I), \Gamma(Op), G, f)$ is equivalent to \mathcal{P} .*

Proof Consider a sequence of steps $Z = \langle O_1, O_2, \dots, O_n \rangle$ with $O_i \in Op$, and associate it with the sequence $Z' = \langle \gamma(O_1), \dots, \gamma(O_n) \rangle$ of operators $\in \Gamma(Op)$. This association represents a *bijection*, as the transformation γ is $1-1$, and each operator in $\Gamma(Op)$ has been generated by one (and only one) operator in Op .

We need to prove that if Z solves \mathcal{P} , then Z' solves \mathcal{P}' , and vice versa. Let

$$I \xrightarrow{O_1} S_1 \xrightarrow{O_2} S_2 \xrightarrow{O_3} \dots \xrightarrow{O_n} S_n = F$$

be the sequence of states obtained applying the plan Z , and

$$\mathbf{C}_R(I) \xrightarrow{\gamma(O_1)} S'_1 \xrightarrow{\gamma(O_2)} \dots \xrightarrow{\gamma(O_n)} S'_n = F'$$

the sequence generated by Z' .

Because of the property 7, since $\mathbf{C}_R(I) \xrightarrow{\gamma(O_1)} S'_1$, then $S'_1 = \mathbf{C}_R(S_1)$. Analogously, $S'_2 = \mathbf{C}_R(S_2)$, $S'_3 = \mathbf{C}_R(S_3)$, \dots , $S'_n = \mathbf{C}_R(S_n)$. Let us see that if $S_n = F$ is a *final* state for \mathcal{P} (i.e. it contains the goal set G) then $S'_n = F' = \mathbf{C}_R(S_n)$ is a final state for \mathcal{P}' , and vice versa.

F is final state for \mathcal{P} iff

$$\forall g \in G, \quad g \in LA^*(F) \quad (8)$$

by definition of correct plan solution for a LA-planning problem. This is equivalent to

$$G \subseteq f(\mathbf{C}_R(F))$$

But since $G' = G$, this condition is the same as

$$G' \subseteq f(\mathbf{C}_R(F))$$

which becomes

$$G' \subseteq f(F') \quad (9)$$

This condition expresses the fact that F' is a final state for the new problem \mathcal{P}' . Since it results that (8) holds iff (9) holds, then we have proved that if $S_n = F$ is a *final* state for \mathcal{P} then $S'_n = F'$ is a final state for \mathcal{P}' , and vice versa.

The second part of the proof concerns the other requirement which a correct plan solution for a LA-planning problem should satisfy, namely, that for every operator O_i used in a solution, if P is its precondition list, then

$$P \subseteq LA^*(S_i)$$

where S_i is the state to which O_i is applied. Let us see that if the operator O_i has preconditions $P \subseteq LA^*(S_i)$, then the corresponding operator $\gamma(O_i)$ has preconditions $P' \subseteq f(S'_i)$, and vice versa.

The condition

$$P \subseteq LA^*(S_i) \quad (10)$$

is equivalent to

$$P' \subseteq f(\mathbf{C}_R(S_i))$$

as $P = P'$; but this is equivalent to

$$P' \subseteq f(S'_i) \quad (11)$$

Since it results that (10) holds iff (11) holds, then we can conclude that if the operator O_i has preconditions $P \subseteq LA^*(S_i)$, then the corresponding operator $\gamma(O_i)$ has preconditions $P' \subseteq f(S'_i)$, and vice versa.

Since both of the conditions which guarantee the correctness of a plan are equivalent for each pair of sequences Z and Z' , then Z solves \mathcal{P} iff Z' solves \mathcal{P}' . Hence, the problem \mathcal{P} is equivalent to \mathcal{P}' . □

5 Complexity of A1

This section shows that the computational complexity of the algorithm A1 to transform a LA-planning problem (I, Op, G, \mathbf{C}_R) is *linear* in the cardinality of the sets R, I and Op .

The first part of A1 consists of the calculation of $I' = \mathbf{C}_R(I)$. Because of the property of acyclicity of the set R , the complexity of this calculation is simply *linear* in the number of propositions present in the set I . In fact, the closure $\mathbf{C}_R(I)$ can be calculated directly as $I \cup K$, where

$$K = \{c \in Lo \setminus Li \mid \exists r \in R : r \equiv (p \vdash c), p \in I\}$$

The set K contains only the collection of the consequences of the LAs which have their premiss already in I , for the addition of K to the closure will not ‘activate’ any other axiom.

Notice that the coefficient of proportionality which relates the number ‘ N_I ’ of operations necessary to calculate the closure of I to the number of propositions in I contains the cardinality of the set of rules R . In other words, there exists a constant ‘ k ’ such that

$$N_I \leq k \cdot (\#R) \cdot (\#I)$$

The second and more important part of the algorithm A1 applies the transformation Γ to the set of operators Op . For each operator $O = (P, A, D) \in Op$ this procedure calculates A' as $\mathbf{C}_R(A)$ and then appends (in two separate steps) a list of conditional deletions to O . The length ‘ l ’ of this list results to be

$$l \leq (\#D) + (\#D) \cdot (\#\mathbf{C}_R(\{d\}) - 1) \leq (\#D) \cdot (\#R + 1)$$

with $d \in D$, as $\#\mathbf{C}_R(S) \leq (\#R) + 1$ if S contains only one element and R is a set of acyclic LAs. Since this calculation is performed for each operator, the computational load N_{Op} required for the entire set Op of operators can be estimated as

$$N_{Op} \leq k' \cdot (\#A^* + \#D^*) \cdot (\#R) \cdot (\#Op)$$

where A^* and D^* are the longest ‘add’ and ‘delete’ lists present in the operators. Assuming $\#A^*$ and $\#D^*$ to be smaller than a certain constant, by adding the N_I to N_{Op} we obtain the overall computational load of A1:

$$N = N_I + N_{Op} \leq k'' \cdot (\#R) \cdot (\#Op + \#I)$$

which shows the linear nature of the computational complexity of A1. Notice that the constant of proportionality k'' grows linearly with the maximum number of propositions in the A and D lists of the operators.

6 Relaxation of Acyclicity

Procedure A1 relies on the hypothesis that the set R of LAs which are pre-processed satisfies the condition of acyclicity. However, this strong requirement does not have to hold ‘globally’ for R . In fact, let us suppose that the function LA^* can be *decomposed* into many separate functions:

$$LA^* = f_n \circ f_{n-1} \circ \dots \circ f_2 \circ f_1$$

where each component f_i calculates a closure C_{R_i} using a specific acyclic set of axioms R_i . Then, instead of pre-processing the function LA^* into the problem \mathcal{P} in one single step, the transformation could be split into n applications of A1: the first one will pre-compile the function f_1 , producing an *equivalent* problem with a new outer extension function

$$LA_1^* = f_n \circ f_{n-1} \circ \dots \circ f_2$$

The second will integrate f_2 , producing a new problem with

$$LA_2^* = f_n \circ f_{n-1} \circ \dots \circ f_3$$

until, after n similar steps, the last application of A1 to f_n will lead to the final LA-planning problem with $LA^* = Id$, which can be solved by a standard planning system.

In order to decompose the function $LA^* = C_R$ into a ‘sequence’ of separate closures it is necessary to split the set of axioms R into an ordered list of subsets R_1, R_2, \dots, R_k such that $\forall i \in \{1, \dots, k\}$ the *consequences of R_i do not appear in the premisses of R_1, \dots, R_{i-1}* , with $\bigcup_i R_i = R$.

If this is possible, then the function C_R can be re-written as

$$C_R = C_{R_n} \circ C_{R_{n-1}} \circ \dots \circ C_{R_1}$$

If each set R_i is also acyclic, this closure function can be pre-processed in n repeated applications of the procedure A1.

Summarising, in order for A1 to be applicable to a LA-planning problem \mathcal{P} with set R , it is sufficient that R can be divided into k subsets R_1, \dots, R_k such that the consequences of R_i do not appear in any of the premisses of R_1, \dots, R_i . Hence, the requirement of global acyclicity of R has been replaced by a weaker condition of *decomposability* into a ‘non-looping’ sequence of subsets.

It should be pointed out that the algorithm A1 transforms a problem \mathcal{P} containing STRIPS-style operators into an equivalent problem \mathcal{P}' containing operators with conditional effects. If A1 needs to be re-applied to its own output, the procedure has to be generalised so that A1 accepts in input also problems containing conditional operators.

The pre-processing of a conditional operator $[(P, A, D), e_1, \dots, e_k]$ can be realised by applying algorithm A1 to the triple (P, A, D) as if it was a normal operator, and then re-applying it to each of the conditional effects (P_i, A_i, D_i) , for $i = 1 \dots k$. This approach, however, can lead to the generation of operators with longer and longer lists of conditional effects. In alternative, the output of A1 could be transformed into an equivalent LA-planning problem which does not contain conditional operators by applying, for example, the method described by Gazen and Knoblock in [5].

7 Conclusions and Future work

The idea underlying the previous analysis, based on the distinction between inner and outer languages for the description of a specific planning domain, seems to be applicable to many different situations. For example, an implementation of A1 was realised and tested in the domain of discourse planning, and produced valid examples of plans for *persuasive discourses* (cf. [1]).

The formal distinction between inner and outer language enables a simple state-definition language to co-exist with an expressive language for the definition of the operators and goals of a planning problem. Although the introduction of an extended and more expressive outer language involves the use, in the problem definition, of language axioms, the algorithm presented transforms automatically a planning problem with LAs into an equivalent one with conditional operators and no LAs. Such algorithm has been shown to be correct and to present *linear* computational complexity.

The main advantages of this approach to the problem of language (or domain) axioms and, more in general, to the issues introduced by the use of more expressive languages are that it is not necessarily specific to one planner and that it does not require any modification of the code and structure of the planning system adopted. The limits of the solution presented lie mainly in the hypotheses on which A1 relies, namely, the acyclicity and ‘atomicity’ of the premiss of the LAs. A possible extension of this approach to deal with language domains which present cyclic and multiple-premisses LAs is sketched in [4].

References

- [1] Garagnani, M., Long, D.P., Fox, M. (1998) “Belief Systems for Conflict Resolution”, *Proceedings of the ECAI-98 Workshop on Conflicts Among Agents*, Brighton, England.
- [2] Garagnani, M. (1998) “Belief Systems and Plans for Communication”, *Proceedings of the 15th International Congress on Cybernetics*, Namur, Belgium.

- [3] Garagnani, M. (1998) "Converting Inference Rules into Conditional Effects", *Proceedings of the 17th Workshop of the UK Planning and Scheduling SIG*, Huddersfield, England.
- [4] Garagnani, M. "Improving the efficiency of Pre-processed Domain-Axiom planning", submitted to *PLANSIG-99* as short paper.
- [5] Gazen, B.C., Knoblock, C.A. (1997) "Combining the Expressivity of UCPOP with the Efficiency of Graphplan", *Proceedings ECP-97*, Toulouse, France.
- [6] Konolige, K (1986) *A Deduction Model of Belief*, Pitman Publishing, London.