

# Frugal Propositional Encodings for Planning\*

Amol D. Mali, Subbarao Kambhampati

Dept. of computer science & Engg.  
Arizona state university, Tempe, AZ 85287-5406  
{amol.mali, rao}@asu.edu

## Abstract

Casting planning as a satisfiability problem has recently lead to the development of significantly faster planners. One key factor behind their success is the concise nature of the encodings of the planning problems. This has lead to an interest in designing schemes for automated generation of propositional encodings for the planning problems. These encodings are inspired by traditional planning algorithms like state space and causal link planning. We examine the existing schemes for generating encodings for causal planning and suggest several improvements to them. We show that these improvements reduce the number of clauses and/or variables in the encoding. Since the number of clauses and variables are related to the hardness of solving a constraint satisfaction problem, our results provide important directions in efficiently encoding planning problems. Our improvements derive plan encoding that is smaller than all other previously developed plan encodings.

## 1 Introduction

A classical planning problem is specified as a 3-tuple  $\langle I, G, A \rangle$ , where  $I$  is completely specified initial state of the world,  $G$  is partly specified goal state, and  $A$  is a set of primitive actions. Each action has preconditions and effects. An action can be applied in a world state only if its preconditions are satisfied by the world state. The effects of the action are guaranteed to be true in the new world state resulting from application of the action, as per classical planning assumption. The planning problem involves finding a sequence of steps (each of which is mapped to an action belonging to the set  $A$ ), such that the sequence can be executed from initial state  $I$  and leads to the goal state  $G$ .

Traditional planning algorithms based on split & prune search have been shown to be very inefficient. [Kautz & Selman 96] reported a satisfiability-based planner that did not use any heuristic knowledge and

yet performed orders of magnitude faster than the other existing planners.

Since solving a planning problem means finding a sequence of actions that is executable from an initial state and leads to goal state, one can cast this problem as a deductive theorem proving problem. One has to prove that there exists a sequence of steps  $[a_1 a_2 a_3 \dots a_n]$  such that  $(I \Rightarrow prec(a_1)), Result(a_1 a_2 a_3 \dots a_n, I) \Rightarrow G$  (where  $prec(a_i)$  denotes the preconditions of step  $a_i$ ), ...,  $Result(a_1 a_2 \dots a_{n-1}, I) \Rightarrow prec(a_n)$ .  $Result(a_1 a_2 a_3 \dots a_i, I)$ , denotes the situation prevailing after the execution of the step sequence  $[a_1 a_2 a_3 \dots a_i]$  from  $I$ . Since such an attempt did not perform well, planning as theorem proving was considered to be infeasible. However these constraints can be converted into a set of clauses and a logical formula can be generated for a given planning problem  $\langle I, G, A \rangle$ . There will be a solution of  $n$  or less non-null actions for the given planning problem if and only if there exists a model of this formula. We refer to this formula in conjunctive normal form (CNF) as a *plan encoding*. Such an encoding can be automatically generated. [Kautz & Selman 92] suggested that by casting planning as a model finding problem, one can exploit the efficient satisfiability solvers like GSAT [Selman, Levesque & Mitchell 92]. Indeed, [Kautz & Selman 96], [Bayardo & Schrag 97] reported very successful results on planning as satisfiability, using stochastic search and a variation of systematic search enhanced with relevance bounded learning respectively. The satisfiability-based planner of [Kautz & Selman 96] is believed to be the fastest of all classical planners currently available.

State space encodings of planning problems have been tested in previous literature [Kautz & Selman 96]. Plan space (causal) encodings have been proposed in [Kautz, McAllester & Selman 96]. However we found that these encodings contain redundant constraints and the encodings can be made more compact. This lead us to propose novel variations of the plan space encodings, each of which expresses the planning

---

In the working notes of the AIPS-1998 workshop "Planning As Combinatorial Search", Pittsburgh, pp. 89-94.

constraints with more parsimony and makes the encodings more economical, either by reducing the number of clauses or variables or both.

[Kautz & Selman 96] and [Kautz, McAllester & Selman 96] point out the importance of reducing the sizes of the encodings. Though a reduction in the number of clauses and/or variables does not universally imply a reduced time to solve the problem, achieving such a reduction has been believed to have payoffs. The empirical results of [Kautz & Selman 96] do show that smaller encodings needed smaller solving times. Various encodings that we developed have different number of clauses and variables. Since these factors affect the hardness of solving an encoding, it is important to find the scheme that leads to as small an encoding as possible. By creating a smaller encoding, we mean reducing the number of clauses, keeping the number of variables constant, or reducing the number of variables, keeping the number of clauses constant, or reducing both the number of clauses and variables. In section 2, we provide five variations of the causal planning encoding of [Kautz, McAllester & Selman 96] and explain them. The variations we propose to the causal encodings preserve soundness and completeness. We discuss the quantitative improvements obtained by applying our variations in section 3 and present conclusions in section 4.

## 2 Revising & Improving the Causal Encodings

Before discussing our variations, we explain the notation here.  $p_i$  denotes a step in the plan.  $k$  denotes the total number of steps in the plan encoding.  $O$  denotes the set of ground primitive actions in the domain. We assume that a primitive action  $o_i, 1 \leq i \leq |O|$ , has  $A_i$  add effects,  $D_i$  delete effects and  $R_i$  preconditions. We call our steps and actions primitive since we do not deal with hierarchical task networks (HTNs). The conversion of an HTN planning problem into a satisfiability problem has been addressed in [Mali & Kambhampati 98].

The add effects of an action  $o_i$  are denoted by  $a_{i1}, a_{i2}, \dots, a_{iA_i}$ , the delete effects are denoted by  $d_{i1}, d_{i2}, \dots, d_{iD_i}$  and the preconditions are denoted by  $n_{i1}, n_{i2}, \dots, n_{iR_i}$ . These are all predicate symbols. The encoding generating schemes assume that a  $k$  step plan exists. We require no-ops here, since the solution may have fewer than  $k$  non-null actions. No-ops (denoted by  $\phi$ ) are null actions with null preconditions and null effects. “ $\prec$ ” denotes temporal precedence relation and “ $p_i \xrightarrow{f} p_j$ ” denotes a causal link from step  $p_i$  to step  $p_j$ . This means that step  $p_i$  has the add effect  $f$ ,  $p_j$  has  $f$  in the list of its preconditions and  $p_i \prec p_j$ .  $p_i, p_j$

are also called contributor and consumer respectively. This link is said to be threatened when there is a step  $p_s$  which can come between  $p_i$  and  $p_j$  and has  $f$  in the list of its *delete* effects. The initial state is assumed to be  $(l_1 \wedge l_2 \wedge l_3 \dots \wedge l_{|I|})$ . The goal state is assumed to be  $(a_1 \wedge a_2 \wedge \dots \wedge a_h)$ .  $\Gamma$  denotes the set of preconditions of all actions. Ground encodings use instantiated actions, as opposed to variablized actions.

In each subsection below, we first discuss what we vary in each encoding. We then provide an explanation of the encoding schemas (that capture the constraints to be satisfied to yield a valid plan) and report the asymptotic size of the encoding. We give formal descriptions of the schemas in the figures.

### 2.1 Eliminating Actions

In the ground causal encoding of [Kautz, McAllester & Selman 96], steps are mapped to actions and as usual, actions have preconditions and effects. Hence actions serve as *middle agents* between steps and preconditions, effects. We eliminate the middle agents here. This change reduces the number of variables. Let us say that there is a step  $p_3$  that can be mapped to any of the available actions, two of which are  $o_1$  and  $o_2$ . Let the *add* effects of  $o_1$  be  $a_{11}, a_{12}$  and the preconditions be  $n_{11}, n_{12}$ . Let the *add* effect of  $o_2$  be  $a_{21}$  and precondition be  $n_{21}$ . Let us assume that these actions do not have any delete effects. [Kautz, McAllester & Selman 96] represent this in the following manner -

$$\begin{aligned} & ((p_3 = o_1) \vee (p_3 = o_2) \vee \dots), (Adds(p_3, a_{11}) \iff \\ & ((p_3 = o_1) \vee \dots)), (Adds(p_3, a_{12}) \iff ((p_3 = o_1) \vee \dots)), \\ & (Needs(p_3, n_{11}) \iff \\ & ((p_3 = o_1) \vee \dots)), (Needs(p_3, n_{12}) \iff ((p_3 = \\ & o_1) \vee \dots)), (Adds(p_3, a_{21}) \iff ((p_3 = o_2) \vee \dots)), \\ & (Needs(p_3, n_{21}) \iff ((p_3 = o_2) \vee \dots)), \end{aligned}$$

We instead say -

$$((Adds(p_3, a_{11}) \wedge Adds(p_3, a_{12}) \wedge Needs(p_3, n_{12}) \wedge Needs(p_3, n_{11})) \vee (Adds(p_3, a_{21}) \wedge Needs(p_3, n_{21}))) \vee \dots$$

Our description clearly has fewer variables. [Kautz, McAllester & Selman 96] mention that actions can be eliminated from the state space encodings for planning. We show that this idea can be exploited in causal encodings as well.

The formal description of the important schemas is provided in Figure 1. We explain the schemas here. Schema 1 captures the fact that a step can be mapped to any of the actions. However we do not refer to the actions, rather we refer to the effects and preconditions of the actions. Schema 2 captures the fact that a step cannot be mapped to more than one action. However as in schema 1, we do not refer to actions directly, we

refer to them through their preconditions and effects. A step cannot need preconditions of more than one action, it cannot add *add* effects of more than one action, and it cannot delete *delete* effects of more than one action. We generate these exclusion clauses only for those pairs of actions for which none of the following three cases hold.

**a.** The effects of  $o_i$  and  $o_j$  are the same, let us say  $(a \wedge b \wedge \neg c)$ . (This representation of effects can be converted into *add* and *delete* lists). The preconditions of  $o_i$  are  $e, f, g$  and those of  $o_j$  are  $e, f$ . The preconditions of  $o_i$  subsume those of  $o_j$ .

**b.** In this case the preconditions of the two actions are the same but the effects are different, effects of one action being subsumed by the effects of another.

**c.** Both preconditions and effects of one action are subsumed by the preconditions and effects of another action respectively.

Schema 3 states that each proposition not mentioned in the initial state is false in the initial state. Schema 4 states that all propositions true in the initial state are added by the initial state. Schema 5 says that if a step needs a precondition, there should be a causal link that supports the precondition. Initial state  $I$  can be viewed as a step that has no preconditions and has *Add* effects that are same as the propositions true in the initial state. This step must come before all other steps. Schema 6 says this. The goal state  $G$  can be viewed as a step  $F$  that has preconditions equal to  $G$  and no effects. Every other step has to come before  $F$ . Schema 7 says this. A step cannot precede itself. This is captured in schema 8. The precedence relation on steps is transitive. This is stated in schema 9. Schema 9 also eliminates cycles of length two. If a step threatens a causal link, then the step should be ordered to come before the contributor or after the consumer (demotion or promotion). This is captured in schema 10. Schema 11 says that for each precondition of goal step  $F$ , there should be a causal link that supports the precondition. Schema 12 says that in a causal link, the contributor adds literal that the consumer needs. This encoding contains  $O(k^3 * |\Gamma| + k * m^{|O|})$  clauses (generated by schemas 1, 10) and  $O(k^2 * |\Gamma| + k * |O| * m)$  variables (required by schemas 1, 5), where  $m = \max(\{A_i + D_i + R_i \mid i \in [1, |O|]\})$ .

## 2.2 Imposing a Total Order On the Steps

In the ground causal encoding of [Kautz, McAllester & Selman 96], each step can be mapped to any action. This encoding does not impose a total order on the steps. Instead the encoding contains all possible step orderings. We argue that this is not necessary. If each step can be mapped to any of the ac-

1.

$$\bigwedge_{i=1}^k (\bigvee_{j=1}^{|O|} ((\bigwedge_{s=1}^{A_j} \text{Add}s(p_i, a_{js})) \wedge (\bigwedge_{t=1}^{R_j} \text{Needs}(p_i, r_{jt})))$$

$$\wedge (\bigwedge_{q=1}^{D_j} \text{Dels}(p_i, d_{jq})) \vee (p_i = \phi))$$

2.

$$\bigwedge_{i=1}^k \bigwedge_{j=1}^{|O|} \bigwedge_{z=1, z \neq j}^{|O|} \neg(((\bigwedge_{s=1}^{A_j} \text{Add}s(p_i, a_{js})) \wedge (\bigwedge_{t=1}^{R_j} \text{Needs}(p_i, r_{jt})))$$

$$\wedge (\bigwedge_{q=1}^{D_j} \text{Dels}(p_i, d_{jq}))) \wedge$$

$$((\bigwedge_{s=1}^{A_z} \text{Add}s(p_i, a_{zs})) \wedge (\bigwedge_{t=1}^{R_z} \text{Needs}(p_i, r_{zt})) \wedge (\bigwedge_{q=1}^{D_z} \text{Dels}(p_i, d_{zq}))),$$

$$\bigwedge_{i=1}^k \bigwedge_{j=1}^{|O|} \neg(((\bigwedge_{s=1}^{A_j} \text{Add}s(p_i, a_{js})) \wedge (\bigwedge_{t=1}^{R_j} \text{Needs}(p_i, r_{jt})) \wedge$$

$$(\bigwedge_{q=1}^{D_j} \text{Dels}(p_i, d_{jq}))) \wedge (p_i = \phi))$$

Figure 1: Schemas for causal encoding that do not refer to actions

tions, then one can impose a total order on the steps to make the encoding constrained (without losing soundness and completeness). If there is a solution to the problem, it can be found out by making appropriate step-action mappings. Consider an example. Let there be four actions  $o_1, o_2, o_3$  and  $o_4$  in the domain. Consider an encoding generated to find a 3 step plan. [Kautz, McAllester & Selman 96] represent all possible orderings among the 3 steps  $p_1, p_2, p_3$  (these are  $p_1 \prec p_2, p_2 \prec p_1, p_1 \prec p_3, p_3 \prec p_1, p_2 \prec p_3, p_3 \prec p_2, p_1 \prec p_1, p_2 \prec p_2, p_3 \prec p_3$ ). This is clearly expensive since this leads to a higher number of clauses and variables. They also represent all possible step action mappings. Let us assume that  $[p_2 p_1 p_3]$  is a solution, with the step action mapping  $p_2 = o_1, p_1 = o_3, p_3 = o_4$ . We impose a total order on the steps and the only ordering relations we generate are  $p_1 \prec p_2, p_1 \prec p_3, p_2 \prec p_3$ . If  $[o_1 o_3 o_4]$  is the solution, it can be obtained by finding the step-action mapping  $p_1 = o_1, p_2 = o_3, p_3 = o_4$ .

This variation reduces the number of variables as well as the number of clauses, because we are no longer required to represent all possible orderings among steps. Also, due to total order, we have to represent fewer causal links and fewer threat resolution-related clauses. For example, if there are 10 steps  $p_1, p_2, \dots, p_{10}$ , in an encoding, then a precondition required by step  $p_5$  can be supplied only by either of  $p_1, p_2, p_3, I$  and  $p_4$ . Similarly if any of these causal

links is threatened, it will be threatened by only that step which can come before  $p_5$  and after the contributor.

By imposing a total order on the plan steps we do not lose the causal nature of the encoding. Though we impose total order, we continue to have causal links and threat resolution axioms (though they are stated in a different way). We also want to point out that imposing total order in the causal encoding does not yield the state space encoding. The causal nature of the encoding is preserved. We do not eliminate action variables in this encoding.

The important relevant schemas are shown in Figure 2. Schema 1 says that a step can be mapped to any of the available actions including no-ops. Schema 2 captures the constraint that a step cannot be mapped to more than one action. Schema 5 says that if a step adds a particular literal, the step must have been mapped to an action that adds that literal. Schema 6 mentions that if a step deletes a particular literal, the step must have been mapped to an action that deletes that literal. Schema 7 specifies that if a step needs a particular literal, the step must have been mapped to an action that needs the literal. We impose a total order on the steps. This is stated in schema 11. Schema 12 states that if a step  $p_q$  threatens a causal link, then some step  $p_t$  coming after  $p_q$  and before the consumer must add it back. Because of total order we cannot reorder steps by promotion or demotion to resolve the threats. Schema 12 also says that a causal link should not be threatened by the step coming immediately before the consumer.

Schemas that are not explained or shown in the figure are similar to the schemas in the previous variation. This encoding contains  $O(k^3 * | \Gamma | + k * | O |^2)$  clauses (generated by schemas 2, 12) and  $O(k^2 * | \Gamma | + k * | O | * m)$  variables (from schemas 5,6,7,8) where  $m = \max(\{A_i + D_i + R_i \mid i \in [1, | O |]\})$ .

### 2.3 Imposing Total Order and Eliminating Actions

Here we provide a hybrid scheme that combines ideas from the two preceding schemes. The two previous schemes independently eliminate action representation and partial ordering on steps. Since these changes do not interfere, they can be applied at the same time. In this variation of the encoding, we apply these changes simultaneously. The formal versions of the schemas for generating this encoding have not been shown because these schemas can be obtained by combining the schemas from the encodings in 2.1 and 2.2 appropriately.

This scheme reduces the number of variables by a

8.

$$\bigwedge_{i=1}^k \bigwedge_{j=1}^{|\Gamma|} (Needs(p_i, \gamma_j) \Rightarrow (\bigvee_{q=1, q \neq i}^{i-1} (p_q \xrightarrow{\gamma_j} p_i) \vee (I \xrightarrow{\gamma_j} p_i)))$$

11.

$$\bigwedge_{i=1}^k \bigwedge_{j=i+1}^k (p_i \prec p_j)$$

12.

$$\bigwedge_{i=1}^k \bigwedge_{j=i+1}^k \bigwedge_{q=i+1, q \neq j}^{j-2} \bigwedge_{s=1}^{|\Gamma|} ((p_i \xrightarrow{\gamma_s} p_j \wedge Dels(p_q, \gamma_s)) \Rightarrow (\bigvee_{t=q+1}^{j-1} Adds(p_t, \gamma_s))),$$

$$\bigwedge_{i=1}^k \bigwedge_{j=i+2}^k \bigwedge_{s=1}^{|\Gamma|} (p_i \xrightarrow{\gamma_s} p_j \Rightarrow \neg Dels(p_{j-1}, \gamma_s))$$

Figure 2: Schemas for causal encoding with total order on steps.

higher amount than the two schemes applied separately. This reduction is achieved because of elimination of actions, need for fewer causal links, fewer precedence orderings and fewer threat resolution axioms.

This encoding contains  $O(k * m^{|O|} + k^3 * | \Gamma |)$  clauses and  $O(k * | O | * m + k^2 * | \Gamma |)$  variables, where  $m = \max(\{A_i + D_i + R_i \mid i \in [1, | O |]\})$ .

### 2.4 Eliminating Actions and Using Precondition-Effect Transitions

Since the preconditions of an action imply its effects, if a step  $p_i$  is mapped to an action  $o_j$ , preconditions of  $p_i$  (same as those of  $o_j$ ) imply the add and delete effects of  $p_i$  (same as the add and delete effects of  $o_j$ ). This forms another basis for eliminating the representation of actions. Let step  $p_2$  be mapped to any of the domain actions one of which is  $o_4$ . Let the *add* effects of  $o_4$  be  $a_{41}, a_{42}$  and *delete* effects be  $d_{41}, d_{42}, d_{43}$ . Let the preconditions of  $o_4$  be  $n_{41}, n_{42}$ . The portion of causal encoding of [Kautz, McAllester & Selman 96] which is immediately relevant to this case, says that,

$$\begin{aligned} ((p_2 = o_4) \vee \dots), (Adds(p_2, a_{41}) \iff ((p_2 = o_4) \vee \dots)), \\ ((p_2 = o_4) \vee \dots), (Adds(p_2, a_{42}) \iff ((p_2 = o_4) \vee \dots)), \\ (Dels(p_2, d_{41}) \iff ((p_2 = o_4) \vee \dots)), \\ ((p_2 = o_4) \vee \dots), (Dels(p_2, d_{42}) \iff ((p_2 = o_4) \vee \dots)), \\ ((p_2 = o_4) \vee \dots), (Dels(p_2, d_{43}) \iff ((p_2 = o_4) \vee \dots)), \\ ((p_2 = o_4) \vee \dots), (Needs(p_2, n_{41}) \iff ((p_2 = o_4) \vee \dots)), \\ (Needs(p_2, n_{42}) \iff ((p_2 = o_4) \vee \dots)), \end{aligned}$$

This means that the step  $p_2$  can be mapped to any of the available actions. Also, if  $p_2$  adds a certain predicate,  $p_2$  must have been mapped to an action that has the predicate in the list of *add* effects. If  $p_2$  deletes a certain predicate,  $p_2$  must

1.

$$\begin{aligned} & \bigwedge_{i=1}^k (\bigwedge_{j=1}^{|O|} ((\bigwedge_{s=1}^{R_j} Needs(p_i, r_{js}))) \iff \\ & ((\bigwedge_{q=1}^{A_j} Adds(p_i, a_{jq})) \wedge (\bigwedge_{t=1}^{D_j} Dels(p_i, d_{jt}))) \vee (p_i = \phi)) \end{aligned}$$

Figure 3: Schemas for causal encodings that do not refer to actions and use state transitions.

have been mapped to an action that has the predicate in the list of *delete* effects. If  $p_2$  needs a certain predicate,  $p_2$  must have been mapped to an action that has the predicate in the list of its preconditions. We avoid this description. Rather we just say  $((Adds(p_2, a_{41}) \wedge Adds(p_2, a_{42}) \wedge Dels(p_2, d_{41}) \wedge Dels(p_2, d_{42}) \wedge Dels(p_2, d_{43})) \iff (Needs(p_2, n_{41}) \wedge Needs(p_2, n_{42}))) \wedge \dots$  It can be verified that this modification reduces the number of clauses and variables. The relevant schema is shown in Figure 3. This scheme reduces the number of variables as well as the number of clauses, because of the use of state transitions and elimination of action representation.

Schema 1 directly assigns preconditions and effects to steps. It says that if preconditions of a step are true, its effects will be true. It allows a step to have effects and preconditions of any action. It captures the fact that a step can be mapped to any action. We need to rule out unintended models. Hence we also specify that if a step adds the *add* effects of an action and deletes the *delete* effects of an action, then the step must need the preconditions of that action.

Note that the schema in Figure 3 applies to the case where no two actions in the domain have same preconditions or same effects. If two or more actions have the same preconditions, we need to say that if a step needs the preconditions of one of these, the step must have effects of one of these actions. If two or more actions have the same effects, we need to say that if a step has the effects of one of these, the step needs preconditions of one of these actions. Other schemas are similar to the schemas in the previous variations.

This encoding has  $O(k^3 * |\Gamma| + k * |O|^2)$  clauses and  $O(k * m * |O| + k^2 * |\Gamma|)$  variables, where  $m = \max(\{A_i + D_i + R_i \mid i \in [1, |O|]\})$ .

## 2.5 Using State Transitions and Imposing Total Order

We impose total order on the steps in the encoding from 2.4 to derive this encoding. This scheme preserves the advantages of both state transitions and total order. Hence it reduces the number of variables and clauses by a higher amount than the two schemes

applied separately. The reduction is achieved because of the use of state transitions, elimination of actions, representation of fewer causal links, fewer precedences and fewer threat resolution axioms. The schemas for this encoding can be derived from the schemas for the encodings in 2.2 and 2.4. This encoding has  $O(k^3 * |\Gamma| + k * |O|^2)$  clauses and  $O(k * |O| * m + k^2 * |\Gamma|)$  variables, where  $m = \max(\{A_i + D_i + R_i \mid i \in [1, |O|]\})$ .

## 3 Quantitative Improvements

We discuss here the improvements that we obtain over the ground causal encoding of [Kautz, McAllester & Selman 96]. The reduction in the number of clauses and variables is reported below. These expressions are an algebraic difference between the sizes of our encoding and the ground causal encoding of [Kautz, McAllester & Selman 96].

Eliminating the representation of actions in an encoding (as in Figure 1, variation 1) reduces the number of variables by  $(k * |O|)$ . This however increases the number of clauses. Imposing the total order on steps of an encoding (as in Figure 2, variation 2) reduces the number of variables by

$$k + \frac{k^2}{2} * (1 + |\Gamma|) - \frac{k}{2} * (1 + |\Gamma|)$$

and the number of clauses by

$$(k + k * (k-1) * (k-2) + (k-1) * |\Gamma| * \frac{(5 * k^2 - k + 18)}{6}) +$$

$$\frac{(k-1) * (k+2)}{2} * (h + |I|) - (k-1) * |I| - h$$

Applying the two variations above simultaneously (variation 3) reduces the number of variables by

$$k + k * |O| + \frac{k^2}{2} * (1 + |\Gamma|) - \frac{k}{2} * (1 + |\Gamma|).$$

The use of state transitions to eliminate the action representation (as in Figure 3, variation 4), reduces the number of variables by  $k * |O|$  and the number of clauses by

$$k + k * \left( \sum_{j=1}^{|O|} (A_j + D_j + R_j) \right)$$

Using state transitions and total order together (variation 5) reduces the number of variables by

$$k + k * |O| + \frac{k^2}{2} * (1 + |\Gamma|) - \frac{k}{2} * (1 + |\Gamma|)$$

and the number of clauses by

$$(2 * k + k * (\sum_{j=1}^{|O|} (A_j + D_j + R_j)) + k * (k - 1) * (k - 2) +$$

$$(k - 1) * |\Gamma| * \frac{5 * k^2 - k + 18}{6} +$$

$$\frac{(k - 1) * (k + 2)}{2} * (h + |I|) - (k - 1) * |I| - h)$$

We show below the percentage reduction in the number of clauses and variables achieved by variation 5 over the ground causal encoding in [Kautz, Selman & McAllester 96] and the actual number of clauses and variables in their encoding. We assume that there are 200 actions in the domain, 30 propositions are true in the initial state, 20 propositions must be true in the goal state and the domain has 400 propositions. We vary the number of primitive steps  $k$ .

k	#Vars	#Clauses	Saving (Variables)	Saving (Clauses)
20	189260	3527530	42.3	72.87
50	1074050	50394445	46.67	81.26
75	2362750	168914895	47.73	82.31
100	4152700	399775970	48.28	82.71

This shows that the savings increase as the number of steps increases. It was found that the saving in the number of variables and clauses obtained by only action elimination was very insignificant. For example, if variation 4 is used (which only eliminates actions), the percentage reduction in the number of clauses and variables is 0.48 and 0.03 respectively. But if we impose total order in addition to action elimination (which is what variation 5 does), the savings increase by orders of magnitude.

The asymptotic size of the causal encoding is determined by the clauses for doing threat resolution and generating causal links. Since the notions of causal links and threat resolution are an inseparable part of causal planning, an improvement in the asymptotic size of the causal propositional encoding appears to be unlikely. One can eliminate the  $O(k^2 * |\Gamma|)$  causal link variables of type  $p_i \xrightarrow{f} p_j$ , but only at the cost of the number of clauses (which increases exponentially). As a result, strategies that achieve a practically useful reduction in the size of an encoding will be important.

It is correctly argued in [Kautz, McAllester & Selman 97] that the value of propositional reasoning depends on our ability to find suitable encodings of practical problems. Our modifications to the causal encoding yield reductions of significant practical value.

## 4 Conclusion

In this work we have critically examined the causal propositional encoding of planning problems and provided more efficient schemes for generating this encoding. In the previously developed encoding, there is a mapping between steps and actions and between actions and preconditions, effects. We have showed that this redundancy can be removed. We have also showed that it is not necessary to represent all possible step orderings and all possible causal links in an encoding, if the encoding allows all step-action mappings. We have showed that these two changes strictly reduce the number of clauses and/or the number of variables. We have showed that these two modifications to the anatomy of an encoding can be applied together, leading to further improvement. Since the number of clauses and variables are related to the hardness of solving an encoding, our results provide important insights into generation of frugal plan encodings.

It is claimed that the lifted version of causal encodings in [Kautz, McAllester & Selman 96] has the smallest size, when compared with the other kinds of plan encodings. However the redundancies that we removed exist in the lifted version as well, indicating a clear scope for further improvement. The removal of the slack in this encoding using our improvements yields plan encodings that are the smallest of all the encodings currently existing. An important issue that our work raises is - what is the minimal encoding? How can one derive such an encoding? Our work has taken useful steps towards answering these questions.

## References

- [Bayardo & Schrag 97] Roberto Bayardo Jr. and Robert Schrag, Using CSP look back techniques to solve real world SAT instances, Proc. of AAAI-97.
- [Kautz, McAllester & Selman 96] Henry Kautz, David McAllester and Bart Selman, Encoding plans in propositional logic, Proc. of KRR-96.
- [Kautz & Selman 92] Henry Kautz and Bart Selman, Planning as satisfiability, Proc. of ECAI-92.
- [Kautz & Selman 96] Henry Kautz and Bart Selman, Pushing the envelope: Planning, Propositional logic and Stochastic search, Proc. of AAAI-96.
- [Kautz, McAllester & Selman 97] Henry Kautz, David McAllester and Bart Selman, Exploiting the variable dependency in local search, Poster session abstract, IJCAI-97.
- [Selman, Levesque & Mitchell 92] Bart Selman, H. Levesque and D. Mitchell, A new method for solving hard satisfiability problems, Proc. of AAAI-92, 440-446.
- [Mali & Kambhampati 98] Amol D. Mali and Subbarao Kambhampati, Encoding HTN planning in propositional logic, Proceedings of AIPS-98.