

On the Hybrid Propositional Encodings of Planning ¹

Amol Dattatraya Mali

Electrical Engg. & Computer Science
University of Wisconsin, Milwaukee, USA

e-mail: mali@miller.cs.uwm.edu

Phone (W): 1-414-229-6762

Fax: 1-414-229-2769

¹Appears in Computational Intelligence Journal, Vol. 18, No. 3, August 2002, pp. 386-419.

Abstract

Recently, casting planning as propositional satisfiability (SAT) has been shown to be an efficient technique of plan synthesis. This paper is a response to the recently proposed challenge² of developing novel propositional encodings that are based on a combination of different types of plan refinements and characterizing the tradeoffs. We refer to these encodings as hybrid encodings. An investigation of these encodings is important, since this can give insights into what kinds of planning problems can be solved faster with hybrid encodings.

Encodings based on partial-order planning and state-space planning have been reported in previous research. We propose a new type of encoding called a *unifying encoding* that subsumes these two encodings. We also report on several other hybrid encodings. Next we show how the satisfiability framework can be extended to incremental planning. State-space encoding is attractive because of its lower size and the causal encoding is attractive because of its highest flexibility in reordering steps. We show that hybrid encodings have a higher size and a lower flexibility in step reordering and, thus, do not combine the best of these. We discuss in detail several specific planning scenarios where hybrid encodings are likely to be superior to non-hybrid encodings.

Keywords: Planning, Propositional Encodings, Satisfiability

²The challenge [Kambhampati 1997] proposed at the recent international joint conference on artificial intelligence.

1 Introduction

A classical planning problem is the problem of synthesizing a sequence of actions from a given set of ground actions such that the action sequence can be executed from a completely described initial state to achieve a goal. Plans are synthesized under the assumptions of deterministic actions, perfect perception and a static and a completely observable environment. Traditional classical refinement planners [Kambhampati 1997(b)] that work in the “split & prune” style have not been very efficient for domain-independent planning. Recently impressive results were obtained by recasting planning problems as propositional satisfiability [Kautz & Selman 1996]. Because of progress in SAT solving, a lot of SAT instances with around 10^4 boolean variables and 10^6 clauses can now be solved [Kautz & Selman 1997(a)] very fast. The satisfiability paradigm has been shown to be capable of synthesizing long plans for some problems with search spaces of sizes around 10^{100} [Kautz & Selman 1997(a)], e.g. plans with 20 to 25 steps, where each step may be bound to any of around 100 actions.

The success of planning as satisfiability has generated interest in developing different propositional encodings of planning problems. These encodings are of interest because they have different sizes that are generally correlated with the ease of finding their models. However larger encodings are not always harder to solve. The size of an encoding is measured in terms of the number of clauses, the sum of the sizes of clauses, and the number of variables in the encoding. A propositional encoding of a planning problem represents the relevant planning constraints (e.g. action choices, pre-conditions and effects of the actions, mutual exclusivity relations between the actions, etc.) in a structure of a fixed size by bounding the number of steps of potential plans. A propositional encoding [Kautz et al. 1996] repre-

sents all possible action sequences of length k and solving it can be viewed as the process of extracting a plan of k steps from this structure [Kambhampati 1997]. The encoding is set up in such a way that it has a model if and only if there is a valid plan. If the assignment of true value to a conjunction of a k step action sequence can be propagated to satisfy a k step encoding without assigning true value to any additional action variable, then the k step action sequence is a plan. Thus, encodings can also be viewed as a means of proving plan correctness [Mali & Kambhampati 1999]. Another representation that compactly encodes action sequences of a certain length (some of which may be plans) is the planning graph constructed by Graphplan [Blum & Furst 1995]. A planning graph cannot be directly fed to a SAT solver. However, a propositional encoding can be formed using information from a planning graph.

Current implementations of planning as satisfiability such as [Ernst et al. 1997],[Kautz & Selman 1992, 1996],[Majercik & Littman 1998], [Kautz & Selman 1998(a)], [Kautz & Selman 1998(b)], [Baiocchi et al. 1998], [Giunchiglia et al. 1998], [Gerevini & Schubert 1998], [Bayardo Jr. & Schrag 1997], [Rintanen 1998], and [McCain & Turner 1998] use only state-space encodings of planning problems. These encodings are based on traditional state-space planning. Empirical evaluation by many of these researchers shows that the state-space encoding with explanatory frame axioms is the smallest and also the fastest to solve. When we refer to smallest/minimal encoding in the rest of the paper, we mean state-space encoding with explanatory frame axioms, unless stated otherwise. An encoding based on partial-order planning is reported in [Kautz et al. 1996] and [Bedrax-Weiss et al. 1996]. It is larger, and harder to solve than the state-space encoding with explanatory frame axioms [Mali &

Kambhampati 1999].

Several challenges on combining ideas from traditional planning with new advances like planning as SAT are posed to the planning community in [Kambhampati 1997]. One of these challenges is to develop and provide analysis for new hybrid propositional encodings, which combine important notions from existing encodings like state-space encoding and partial-order, or causal encoding. Addressing this challenge is important because there are good reasons to form hybrid encodings. For example, state-space encodings are smaller but causal encodings allow more flexibility in reordering steps while modifying plans. Also, certain domain-specific constraints are more easily added to state-space encodings while others are more easily added to causal encodings. If hybrid encodings could be created that have the best of both these worlds, we might be able to solve certain kinds of planning problems faster. This motivates the following investigation.

A key difference between state-space planning and partial-order planning is that the state of the world is represented at each time step during the state-space planning process, but it is never available during the partial-order planning process. This observation also applies to the corresponding encodings. One can know the complete state of the world at each time step just by inspecting the model of a state-space encoding. The model of a causal encoding does not provide the state information. We can bridge the extremes of not representing world states at all versus maintaining world states at each time step by controlling the number of time steps at which the world state is represented. We show that the state-space and partial-order encodings of planning [Kautz et al. 1996], [Mali & Kambhampati 1999], are subsumed by our new encoding called the *unifying encoding*. The state of the world is represented at a

variable number of time steps in the unifying encoding. If the number of time steps between the initial state and the goal at which world states are represented, n , is less than the number of steps in the encoding, k , we get an encoding that is neither a state-space encoding nor a causal encoding. If $n = 0$, the unifying encoding becomes the same as the causal encoding. If $n = k$, the unifying encoding becomes the same as the state-space encoding.

The world states can be viewed as partitioning the set of steps. The subsets of the steps lie in the “regions” between consecutive world states. A region is a set of steps, S , that may be partially ordered such that the partial order relation is defined between each member of S and each step outside S . Causal link symbols and symbols for partial ordering between steps are then used to represent causal links and precedence only between steps in the same region. No causal link symbols and step precedence symbols are used to represent causal links and ordering between steps in different regions. For example, the unifying encoding in Figure 1 has two regions. An instance of the unifying encoding with $1 \leq n < k$ is a hybrid encoding.

To answer the challenge [Kambhampati 1997], we develop several other hybrid encodings, which combine the key notions from state space planning and causal planning. We compare the sizes of these encodings for the domain-independent non-incremental planning scenario (where a plan is synthesized from scratch) and show that none of these hybrid encodings is smaller (has fewer clauses and variables) than the smallest of the currently existing encodings. We then examine the sizes of the current and the hybrid encodings for the generalized-incremental planning scenario and the problem of plan completion. In the generalized-incremental planning scenario, the actions from an old plan may be removed and new actions

may be added. In the problem of plan completion, an incomplete plan is extended to solve a problem without violating any of its constraints. We also compare the sizes of the current (non-hybrid) and the hybrid encodings for the domain-independent incremental planning scenario, when extra clauses are added to these encodings to control the number of occurrences of actions in plans, to control non-minimality of plans.

This work makes the following contributions -

- Synthesis of several hybrid encodings and a comparison of their asymptotic sizes with the current encodings in non-incremental domain-independent planning where plans are synthesized from scratch.
- An adaptation of the current and hybrid encodings to a generalized-incremental planning, the specific problem of plan completion, and a comparison of their asymptotic sizes.
- A comparison of the asymptotic sizes of the encodings (current and hybrid) when extra clauses are added to control non-minimality (redundancy) in the plans found.
- An empirical evaluation of several instances of the unifying encoding which shows that its hybrid instances are neither smaller nor easier to solve than the smallest encoding.
- We also show that in domain-independent non-incremental planning the hybrid encodings do not combine the best of both worlds (lowest size of the state-space encoding with explanatory axioms and the highest flexibility of causal encoding in allowing steps to be reordered).

- Finally, we identify several kinds of specific planning problems where hybrid encodings are smallest or are likely to be smallest.

The rest of the paper is organized as follows. We explain how the current propositional plan encodings differ and introduce the notion of the unifying encoding (in Section 2). At the beginning of this background section we also explain some planning terminology and report on previous work on planning as model finding. We explain the notation and terminology used to describe the constraints in the encodings in Section 3. We show how the unifying encoding can be automatically generated in Section 4. We report the size of the unifying encoding in Section 5. In this section, we also show how the current encodings can be viewed as instantiations of the unifying encoding. In Section 6, we develop several other hybrid encodings and report the number of clauses and variables that they contain. Furthermore, we prove a theorem which states that no hybrid encoding can be smaller than the smallest encoding in domain-independent non-incremental planning. In Section 7, we discuss how the unifying, state-space, and causal encodings can be adapted to handle incremental planning and report their sizes to handle this scenario. In Section 8, we report on empirical evaluations of various instances of the unifying, state-space, and causal encodings from [Kautz et al. 1996]. In Section 9, we show how extra clauses can be used to control non-minimality in plans found by solving the unifying, state-space, and causal encodings. In Section 10, we discuss several kinds of planning problems for which hybrid encodings are the smallest or are likely to be smallest types of encodings. We also supply supporting size-related theoretical analysis. Furthermore, we discuss tradeoffs in the encodings along with directions for future work. We present the conclusions in Section 11.

2 Background

In this section we first explain some terms in classical planning like step, fluent, ground action, ground fluent, arity of an action, and arity of a fluent. We also clarify the distinction between a step and an action, and report on work on planning as model finding. After this, we explain key differences between state-space and causal encodings and show how a unifying encoding can be developed.

2.1 Classical Planning Terminology

A plan is an executable sequence of ground actions. A ground action does not contain variables. For example, “ $\text{move}(x,y,z)$ ” is a variablized action for moving block x from the top of block y to the top of block z . Instantiating variables, we see that $\text{move}(x,A,B)$ where A and B are constants (specific blocks) is a partially instantiated action while $\text{move}(C,A,B)$ is a fully instantiated, or ground, action. The arity of an action is the number of its arguments. The arity of “ $\text{move}(A,B,D)$ ” is 3.

A plan can also be defined as a sequence of steps, where each step is bound to an executable action. The distinction between the terms “step” and “action” is as follows: A step occurs in a plan only once and an action may occur multiple times. For example, if “ $\text{drive}(T,\text{Milwaukee},\text{Chicago})$ ” and “ $\text{drive}(T,\text{Chicago},\text{Milwaukee})$ ” represent the action of driving truck T from Milwaukee to Chicago and the action of driving truck T from Chicago to Milwaukee respectively, these actions may be needed multiple times in plans for transporting multiple packages from Milwaukee to Chicago. This can be seen by considering the case where T is the only truck available and the volume or carrying capacity of the truck is less than

the total volume or weight of all packages to be moved. For example, [load(P1,T,Chicago) drive(T,Chicago,Milwaukee) unload(P1,T,Milwaukee) drive(T,Milwaukee,Chicago) load(P2,T,Chicago) drive(T,Chicago,Milwaukee) unload(P2,T,Milwaukee)] is a plan for transporting packages P1 and P2 from Chicago to Milwaukee. This plan can also be represented as the sequence of steps [p1 p4 p3 p2 p6 p5 p7] where the step-action bindings are p1 = load(P1,T,Chicago), p4 = drive(T,Chicago,Milwaukee), p3 = unload(P1,T,Milwaukee), p2 = drive(T,Milwaukee,Chicago), p6 = load(P2,T,Chicago), p5 = drive(T,Chicago,Milwaukee) and p7 = unload(P2,T,Milwaukee). Each step has an action binding and any two steps can have the same action binding. Furthermore, a step can be bound to multiple non-interfering actions. The notion of a step allows us to distinguish between multiple occurrences of an action.

A fluent is a proposition whose truth can change. For example, “on(A,B)” which represents block A being on top of block B, is a fluent. This is because A could be put on a table, or on top of some other block, making on(A,B) false. On the other hand, “color(A,Red)”, which represents color of block A being red, is not a fluent if there is no action for scratching or repainting A. The terms ground fluent and arity of a fluent are defined in a manner similar to ground action and arity of an action. Note that predicates like “on(A,B)” and “move(B,C,D)” can be rewritten as onAB and moveBCD and treated as propositions.

2.2 Planning as Model Finding

The significant impact of planning as satisfiability on the field of artificial intelligence planning is clear from papers like [Braha & Brafman 1998], [Brafman 1999], [Brafman & Hoos 1999], [Do & Kambhampati 2000], [Do et al. 2000], [Ferraris & Giunchiglia 2000], [Gerevini

& Schubert 2000], [Giunchiglia 2000], [Huang et al. 1999, 2000], [Kautz 2000], [Kautz & Selman 1999], [Mali 1998, 1999(a), 1999(b), 1999(c), 2000(a), 2000(b)], and [Mali & Kambhampati 1998(a), 1998(b), 1998(c)]. Advances in SAT solving are reported in [Crawford & Selman 1996], [Li 2000], and [Li & Anbulagan 1997]. Furthermore, a SAT encoding of a planning problem can be easily converted into a 0-1 ILP encoding (integer linear programming), allowing us to take advantage of even more research. For example, the SAT instance $(x \vee y) \wedge (\neg y \vee z \vee \neg w)$ can be converted into an ILP with the following two constraints: $(x + y) \geq 1, ((1 - y) + z + (1 - w)) \geq 1$, where the domain of variables x, y, z, w is $\{0, 1\}$. Work on planning as ILP is reported in [Bockmayr & Dimopoulos 1999], [Kautz & Walser 1999], and [Vossen et al. 2000]. Work on solving more expressive planning problems that involve boolean and numeric variables (especially when resource quantities are involved) is reported in [Wolfman & Weld 2000]. The solver LPSAT [Wolfman & Weld 2000] combines advances in both SAT solving and LP solving. Empirical results on planning as constraint programming are reported in [van Beek & Chen 1999]. It is clear that planning as constraint satisfaction is an important paradigm and most of the work hitherto carried out in this paradigm is on planning as SAT.

2.3 How the Current Encodings Differ?

Mali & Kambhampati [Mali & Kambhampati 1999] have both theoretically and empirically shown that the causal encoding is strictly larger and harder to solve than the smallest encoding (the state-space encoding with explanatory frame axioms, reported in [Kautz et al. 1996]). Thus, both state-space and causal encodings have been viewed to be radically different. State-space and causal encodings of a specific planning problem are given in [Mali

2001] as an example. In this subsection, we explain the impact of the presence and absence of world state on the sizes of state-space and causal encodings.

In traditional forward state-space planning, a complete description of the world state is available at each time step during the planning process and it is guaranteed that there is a path to reach each of the states. In the context of this encoding, the term “state” has different semantics. The complete and correct world state at each time step is available only after a state-space encoding is solved. This is because a solver may satisfy the clauses in an encoding in an arbitrary order. The world state at time t in the model of a state-space encoding is the conjunction of the truth assignments to the fluents at time t . In state-space encodings, the pre-conditions required for the application of an action at time t are true in the state at time t itself. All changes made to the world by an action applied at time t are hold in the world state at time $(t + 1)$. On the time line, the state is the closest source providing pre-conditions. However, in encodings based on partial-order planning (that we also refer to as *causal encodings*), there is no representation of states. Hence, the pre-conditions of a step may be made available (true) by any set of steps that precede it. Thus, a step has to rely upon the non-interference of an arbitrary number of earlier steps for the satisfaction of its pre-conditions. Since any step that can come between the contributor step and the consumer step can delete the consumer’s supported pre-condition (and hence cause a threat), conflicts have to be resolved by explicitly stating that any clobbering (or threatening) step must be ordered to come before the contributor step or after the consumer step. Hence, the constraints for dealing with all potential interactions between all the steps need to be incorporated in the encoding. This explains the difference between the sizes of

the state-space encodings and the causal encodings.

[Kautz & Selman 1996] use the term “state-based” encoding for an encoding that is based on state-space planning, with the action variables eliminated. Such an elimination is possible because the truths of the pre-conditions and effects of an action are equivalent to the occurrence of the action. We do not insist on the elimination of the action variables. Thus, we call our encoding “state-space encoding”.

2.4 Towards the Unifying Encoding

Our key observation that leads to the development of the unifying encoding is the following: In state-space encodings the closest provider (state) of the pre-conditions of an action, o_i , is zero time steps behind o_i . This is because the world state at time t also provides the pre-conditions. On the other hand, the closest provider of a step’s pre-conditions can be arbitrarily far behind the consumer in causal encodings. If we control the distance between a consumer and the closest contributor by varying the number of time steps at which the world state is represented, we can synthesize a series of encodings. These encodings not only represent world states after time intervals of varying length, but also allow a partial order on the steps that are “sandwiched” between two consecutive world states. By demanding that only one action should occur between two successive states, one gets the state-space linear encoding of [Kautz et al. 1996]. By allowing multiple non-interfering actions to occur at the same time, one gets the parallel state-space encoding of [Kautz et al. 1996]. By removing the representation of states completely, one gets the causal encoding of [Kautz et al. 1996]. This is shown in the example in Figure 1. A unifying encoding is an encoding that contains both world states and causal links. It can be seen that in the unifying encoding, the

interactions among the steps are localized through the partitioning of the set of steps that is achieved by the introduction of world states. The states serve as boundaries between two neighboring regions, where each region contains partially ordered steps. Hence, a step in a region interacts only with other steps from its region rather than all steps in the encoding. It can interact with steps in other regions only indirectly (through a preceding state).

If world states are represented at more time steps while keeping the total number of steps the same, without increasing the number of steps in any region, the amount of localization of interactions is higher. In state-space encoding [Kautz et al. 1996], the amount of localization of interactions is maximum and in their causal encoding, the amount of localization of interactions is minimum. Likewise, the number of causal link variables in state-space encodings [Kautz et al. 1996] is minimum (zero) while the number of causal link variables in causal encoding [Kautz et al. 1996] is maximum. Each step in region i precedes each step in region j , $j > i$. It is important to note that this partitioning into regions preserves the soundness and completeness of planning.

Any instantiation of the unifying encoding (obtained by choosing the number of regions greater than 1 and less than the total number of steps) will be smaller (have fewer clauses and fewer variables) than the causal encoding of [Kautz et al. 1996]. However, any instantiation of the unifying encoding where localization is not maximum will be larger than the state-space encoding.

A plan output by a state-space planner is a sequence of steps. A plan output by a partial-order planner is a set of steps along with the ordering constraints on those steps. A linearization is a sequence of steps that is consistent with the ordering constraints on the

steps in plan. The plan obtained after solving the unifying encoding is a sequence of sets such that each set can be viewed as a local plan containing causal links and partially ordered steps from a particular region. Each local plan has a linearization. A concatenation of any linearizations of the local plans gives a final executable plan. Each local plan linearization used in the final plan must be safe, that is, it must respect the constraints in the local plan.

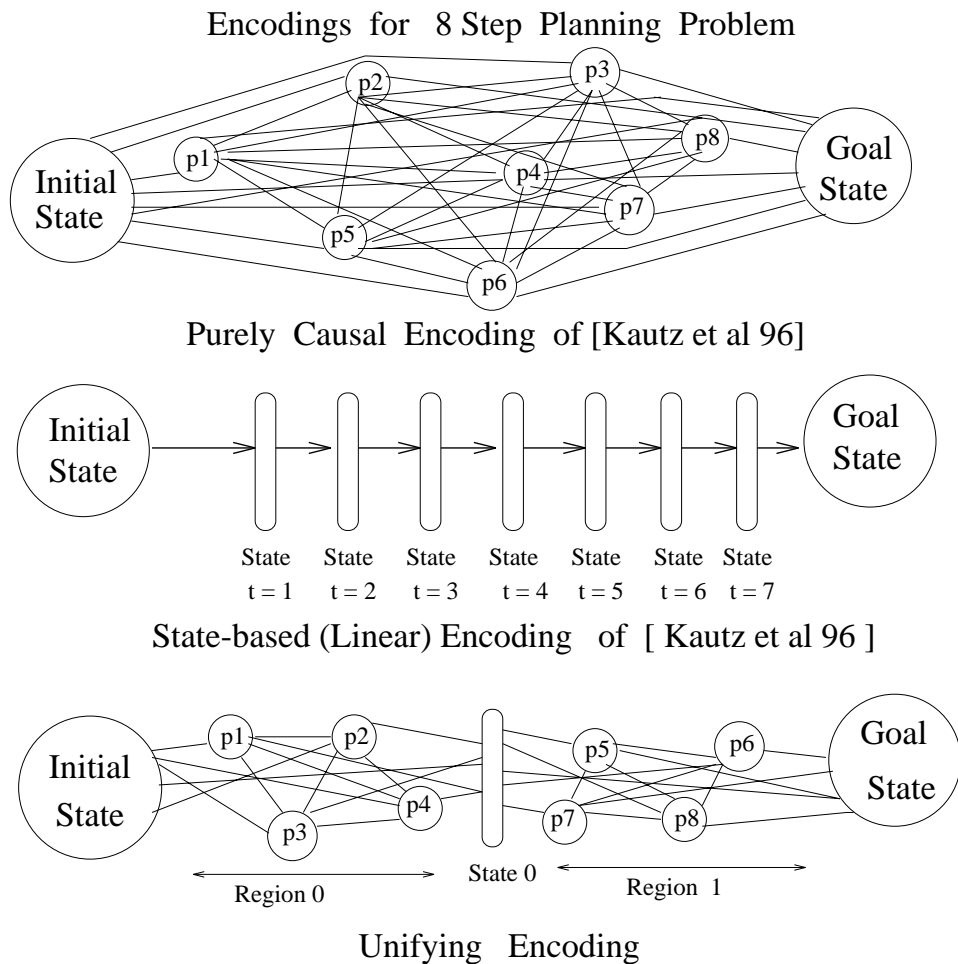


Figure 1: Interactions between steps in propositional encodings of classical planning problems.

3 Notation

In this section, we define several symbols and planning terms. We assume the steps and actions to be primitive since we do not consider hierarchical task networks [Erol et al. 1994],[Kambhampati et al. 1998], or macro operators.

p_i - a primitive step

o_j - a ground primitive action

$o_i(t)$ - occurrence of the action o_i at time step t

k - number of steps in an encoding

U - set of ground unnegated pre-condition and effect propositions

u_j - a ground pre-condition or effect

$u_j(t)$ - u_j at time t

ϕ - null action (no-op)

O - set of all ground actions in the domain.

$(p_i = o_j)$ - step→action mapping or binding.

G - Goal, same as the conjunction $(a_1 \wedge a_2 \wedge a_3 \dots \wedge a_h)$.

a_i - a proposition from goal G

F - the goal step (goal can be viewed as a step with preconditions same as the goal and no effects)

I - conjunction of propositions true in the initial state

l_i - a proposition true in the initial state

$| I |$ - number of propositions true in the initial state

$p_i \xrightarrow{f} p_j$ - causal link where p_i adds the condition f (makes f true) and p_j needs it and p_i precedes p_j

A_j, R_j, D_j - the number of add effects, pre-conditions, and delete effects of the action o_j respectively

a_{js}, r_{jt}, d_{jq} - the individual add effects, pre-conditions, and delete effects of the action o_j respectively

$Adds(p_i, u_j)$ - step p_i having the effect of making u_j true

$Needs(p_i, u_s)$ - u_s being a pre-condition of the step p_i

$Dels(p_i, u_q)$ - step p_i having the effect of making u_q false (deleting u_q)

$p_i * p_j$ - contiguity of the two steps p_i and p_j , so p_i occurs immediately before p_j and no step can occur between p_i and p_j

$p_i \prec p_j$ - the temporal precedence relation (partial order) between the two steps p_i and p_j . p_i occurs before p_j and any number of steps can occur between them.

p - number of regions in a hybrid encoding

m - number of steps in each region of a hybrid encoding, same as $\frac{k}{p}$

$u_q(i) \longrightarrow p_j$ - causal link from world state to a step, where u_q is true in state i and is needed by the step p_j

$p_i \longrightarrow u_j(q)$ - causal link from a step to a world state, where u_j is true in world state q and u_j is made true by step p_i

$u_i(t) \longrightarrow u_i(t+1)$ - causal link between world states, where u_i is true in the world states t and $(t+1)$ and not deleted by any step which occurs between these states

$\neg u_i(t) \longrightarrow \neg u_i(t+1)$ - causal link between world states, where u_i is false in the states t

and $(t + 1)$ and not added (made true) by any step which occurs between these states

q' - the number of steps added to an existing encoding in incremental planning.

The null action does not cause any change to a world state. It is used to be able to find a plan of k' steps by solving a k step encoding when $k' < k$, since the extra $(k - k')$ steps in an oversized encoding can be bound to ϕ .

Note that many symbols in the preceding list are boolean variables. For example, the variable $o_i(t)$ denotes the occurrence of the action o_i at time t . The action o_i will occur in a plan at time t only if the SAT solver used makes the assignment $o_i(t) = true$ in its satisfying truth assignment for the encoding. In that case, $(o_i(t) = true)$, the pre-conditions of o_i are true at time t and its effects are true at time $(t + 1)$. If the SAT solver finds a satisfying truth assignment in which false is assigned to $o_i(t)$, o_i will not occur in the plan at time t . So, the symbol $o_i(t)$ does not mean that o_i necessarily occurs in a plan at time t . Similarly, u_q will be true in the world states t and $(t + 1)$, and won't be deleted by any step between these two states, only if the model of the encoding contains true assigned to the boolean variable $u_q(t) \longrightarrow u_q(t + 1)$.

A “threat” is said to exist if the pre-condition f of a step p_i is made true by a step p_j , $p_j \prec p_i$ and there is a step p_s , such that $p_j \prec p_s, p_s \prec p_i$ and p_s deletes f . \cdot in subscripts and various expressions denotes multiplication.

We assume that there are p regions in the encoding (we use the words “region” and “section” interchangeably). Hence, there are $(p - 1)$ states, between the initial state and the goal, where $1 \leq p \leq k$. Without loss of generality, we assume that all regions contain $m = \frac{k}{p}$ steps. Step indices range from 1 to k and the region indices range from 0 to $(p - 1)$. The

state immediately preceding region 0 is the initial state and the state immediately succeeding region $(p - 1)$ is the goal. The i th region contains steps with indices ranging from $(i.m + 1)$ to $(i + 1).m$. We use the term “minimal encoding” to refer to the encoding that has the fewest number of clauses and variables. We use the term “naive encoding” to refer to an encoding in which each step may be bound to any of the $|O|$ ground actions in the domain. So, a naive encoding contains $O(k |O|)$ variables which represent all step-action bindings.

4 Unifying Encoding

Here we explain the schemas required to generate the unifying encoding (shown in Figure 2). Though we do not separately describe the basics of state-space and causal encodings of [Kautz et al. 1996], the basics as well as the schemas needed to generate these encodings can be derived from the schemas of the unifying encoding. After all, state-space and causal encodings are just special cases of the unifying encoding.

Schema 1 states that each step must be bound to some action. Schema 2 states that a step cannot be bound to more than one action. Schema 3 states that all propositions in the initial state description are true in the initial state while all remaining propositions are false in the initial state. It also requires the goal to be true at time p . Schema 4 states that if a step is mapped to an action o_s (that is not a no-op), the step inherits the preconditions and effects of o_s . Schema 5 states that a step can add, delete, or need a proposition only if it is mapped to an action which adds, deletes, or needs that same proposition.

Schema 6 states that the pre-conditions of each step $p_{i.m+j}$ (this is the j th step in the i th region) must be made available either by some other step in the same region as $p_{i.m+j}$ or by the immediately preceding state. Schema 7 states that any threats to causal links

between steps (in the same region) must be resolved by promotion or demotion (reordering the clobberer). Also, any threats to causal links from the initial state or to the goal must be resolved in a similar manner.

Schema 8 consists of five parts (a, b, c, d and e). Part (a) states that for the causal link $p_{j_1} \xrightarrow{u_q} p_{j_2}$ to exist between two steps p_{j_1} and p_{j_2} both belonging to the same region, p_{j_1} must make u_q true, p_{j_2} must need u_q and p_{j_1} must precede p_{j_2} . Part (b) states that for the causal link $l_q(0) \longrightarrow p_i$ to exist from the initial state to the step p_i , p_i must need l_q and l_q must be true in the initial state. Part (c) states that the causal link $p_i \longrightarrow a_j(p)$ from a step in the region immediately before the goal to the goal state exists only if p_i makes a_j true and a_j is true in the goal. Part (d) states that for the causal link $u_j(i) \longrightarrow u_j(i+1)$ to exist, u_j must be true in both the i th and $(i+1)$ th world states and no step in the region between the i th and the $(i+1)$ th world states deletes u_j . Part (e) states that for the causal link $u_j(i) \longrightarrow p_{i.m+q}$ from the i th world state to the q th step in the i th region, $p_{i.m+q}$, to exist, u_j must be true in the i th world state and $p_{i.m+q}$ must need u_j , and no step preceding $p_{i.m+q}$ in the i th region deletes u_j .

Schema 9 consists of five parts a, b, c, d and e. Part (a) states that if a fluent u_j is true in the i th world state, there is either a causal link from some step in the region immediately preceding the i th world state to $u_j(i)$ or there is some causal link from the $(i-1)$ th world state to the i th world state. Part (b) states that for the causal link $\neg u_q(i) \longrightarrow \neg u_q(i+1)$ to exist, u_q must be false in both the i th and $(i+1)$ th world states and can't be made true by any step in the region between the i th and $(i+1)$ th world states. Part (c) is similar to part (a) with the difference that it contains the negation of u_j . 9(a) and 9(c) are explanatory

frame axioms of the unifying encoding since they explain changes in the truths of fluents. Part (d) states that for the causal link $p_{j_i} \longrightarrow \neg u_q(i+1)$ to exist, u_q must be false in the $(i+1)$ th world state, p_{j_i} must delete u_q , and any step making u_q true must precede p_{j_i} . Part (e) is similar to part (d) with the difference that u_q is true in the $(i+1)$ th world state. Finally, schema 10 states that inconsistent orderings are not allowed. It also enforces the transitivity of the relation \prec on steps.

5 Size of the Unifying Encoding

We now compute the asymptotic number of clauses and variables in the unifying encoding. We also show that the sizes of the encodings from previous literature like [Kautz et al. 1996] can be shown to be the same as sizes of the unifying encoding under certain restrictions. Since each of the k steps from the unifying encoding can be mapped to any of the $|O|$ actions from the domain and, hence, can add, delete, or need any fluent from the set U , a total of $O(k(|O| + |U|))$ variables of various types like $p_i = o_j$, $Adds(p_i, u_q)$, $Needs(p_i, u_s)$ and $Dels(p_i, u_r)$ are needed. Variables of type $p_i \prec p_j$ that represent the partial order on the steps will also be needed. However, since such relations are generated only for steps in the same region, and there are p such regions, the number of such variables will be $O(\frac{k^2}{p})$. A similar argument shows that the number of causal link variables will be $O(\frac{k^2}{p} |U|)$. Ignoring variables that do not dominate the asymptotic size, it can be seen that the number of variables in the unifying encoding is $O(k(|O| + |U|) + \frac{k^2}{p} |U|)$.

Now let us consider the number of clauses. Since a step cannot be bound to more than one action, $O(k |O|^2)$ clauses are required to capture mutual exclusions. Since each of

1. $\bigwedge_{i=0}^{p-1} \bigwedge_{j=1}^m (\bigvee_{s=1}^{|O|} (p_{i.m+j} = o_s) \vee (p_{i.m+j} = \phi))$
2. $\bigwedge_{i=0}^{p-1} \bigwedge_{j=1}^m \bigwedge_{s_1=1}^{|O|} \bigwedge_{s_2=1, s_2 \neq s_1}^{|O|} \neg((p_{i.m+j} = o_{s_1}) \wedge (p_{i.m+j} = o_{s_2}))$
 $\bigwedge_{i=0}^{p-1} \bigwedge_{j=1}^m \bigwedge_{s=1}^{|O|} \neg((p_{i.m+j} = o_s) \wedge (p_{i.m+j} = \phi))$
3. $\bigwedge_{s=1}^{|I|} l_s(0), \quad \bigwedge_{s=1}^{|U-I|} \neg a'_{js}(0), a'_{js} \notin I \quad \bigwedge_{i=1}^h a_i(p)$
4. $\bigwedge_{i=0}^{p-1} \bigwedge_{j=1}^m \bigwedge_{s=1}^{|O|} ((p_{i.m+j} = o_s) \Rightarrow ((\bigwedge_{s'=1}^{A_s} \text{Adds}(p_{i.m+j}, a_{ss'})) \wedge (\bigwedge_{t=1}^{R_s} \text{Needs}(p_{i.m+j}, r_{st})) \wedge (\bigwedge_{q=1}^{D_s} \text{Dels}(p_{i.m+j}, d_{sq}))))$
5. $\bigwedge_{i=0}^{p-1} \bigwedge_{j=1}^m \bigwedge_{q=1}^{|U|} (\text{Adds}(p_{i.m+j}, u_q) \Rightarrow (\bigvee_{s=1}^x (p_{i.m+j} = o_{m_s}))), \text{Adds}(o_{m_s}, u_q)$
 $\bigwedge_{i=0}^{p-1} \bigwedge_{j=1}^m \bigwedge_{q=1}^{|U|} (\text{Dels}(p_{i.m+j}, u_q) \Rightarrow (\bigvee_{s=1}^x (p_{i.m+j} = o_{m_s}))), \text{Dels}(o_{m_s}, u_q)$
 $\bigwedge_{i=0}^{p-1} \bigwedge_{j=1}^m \bigwedge_{q=1}^{|U|} (\text{Needs}(p_{i.m+j}, u_q) \Rightarrow (\bigvee_{s=1}^x (p_{i.m+j} = o_{m_s}))), \text{Needs}(o_{m_s}, u_q)$
6. $\bigwedge_{i=0}^{p-1} \bigwedge_{j=1}^m \bigwedge_{q=1}^{|U|} (\text{Needs}(p_{i.m+j}, u_q) \Rightarrow ((\bigvee_{s=i.m+1, s \neq (i.m+j)}^{m.(i+1)} (p_s \xrightarrow{u_q} p_{i.m+j})) \vee (u_q(i) \longrightarrow p_{i.m+j})))$
7. $\bigwedge_{i=0}^{p-1} \bigwedge_{j_1=1}^m \bigwedge_{j_2=1, j_1 \neq j_2}^m \bigwedge_{j_3=1, j_3 \neq j_2, j_3 \neq j_1}^m \bigwedge_{q=1}^{|U|} (((p_{j_1} \xrightarrow{u_q} p_{j_2}) \wedge \text{Dels}(p_{j_3}, u_q)) \Rightarrow ((p_{j_3} \prec p_{j_1}) \vee (p_{j_2} \prec p_{j_3})))$
 $\bigwedge_{i=1}^m \bigwedge_{q=1}^{|U|} \bigwedge_{j=1, j \neq i}^m (((u_q(0) \longrightarrow p_i) \wedge \text{Dels}(p_j, u_q)) \Rightarrow (p_i \prec p_j))$
 $\bigwedge_{i=m.(p-1)+1}^k \bigwedge_{j=1}^h \bigwedge_{s=m.(p-1)+1, s \neq i}^k (((p_i \longrightarrow a_j(p)) \wedge \text{Dels}(p_s, a_j)) \Rightarrow (p_s \prec p_i))$
8. (a) $\bigwedge_{i=0}^{p-1} \bigwedge_{j_1=(i.m+1)}^{(i+1).m} \bigwedge_{j_2=(i.m+1), j_1 \neq j_2}^{(i+1).m} \bigwedge_{q=1}^{|U|} ((p_{j_1} \xrightarrow{u_q} p_{j_2}) \Rightarrow (\text{Adds}(p_{j_1}, u_q) \wedge \text{Needs}(p_{j_2}, u_q) \wedge (p_{j_1} \prec p_{j_2})))$
(b) $\bigwedge_{i=1}^m \bigwedge_{q=1}^{|I|} ((l_q(0) \longrightarrow p_i) \Rightarrow (l_q(0) \wedge \text{Needs}(p_i, l_q)))$
(c) $\bigwedge_{i=m.(p-1)+1}^k \bigwedge_{j=1}^h ((p_i \longrightarrow a_j(p)) \Rightarrow (\text{Adds}(p_i, a_j) \wedge a_j(p)))$
(d) $\bigwedge_{i=0}^{p-1} \bigwedge_{j=1}^{|U|} ((u_j(i) \longrightarrow u_j(i+1)) \Rightarrow (u_j(i) \wedge u_j(i+1) \wedge (\bigwedge_{q=i.m+1}^{(i+1).m} \neg \text{Dels}(p_q, u_j))))$

$$(e) \bigwedge_{i=0}^{p-1} \bigwedge_{j=1}^{|U|} \bigwedge_{q'=1, q' \neq q}^m \bigwedge_{q=1}^m (u_j(i) \longrightarrow p_{i.m+q} \Rightarrow (u_j(i) \wedge Needs(p_{i.m+q}, u_j) \wedge (Dels(p_{i.m+q'}, u_j) \Rightarrow (p_{i.m+q} \prec p_{i.m+q'}))))$$

$$9. (a) \bigwedge_{i=1}^p \bigwedge_{j=1}^{|U|} (u_j(i) \Rightarrow ((\bigvee_{q=(i-1).m+1}^{i.m} (p_q \longrightarrow u_j(i))) \vee (u_j(i-1) \longrightarrow u_j(i))))$$

$$(b) \bigwedge_{i=0}^{p-1} \bigwedge_{q=1}^{|U|} ((\neg u_q(i) \longrightarrow \neg u_q(i+1)) \Rightarrow (\neg u_q(i) \wedge \neg u_q(i+1) \wedge (\bigwedge_{s=i.m+1}^{(i+1).m} \neg Adds(p_s, u_q))))$$

$$(c) \bigwedge_{i=1}^p \bigwedge_{j=1}^{|U|} (\neg u_j(i) \Rightarrow ((\bigvee_{q=(i-1).m+1}^{i.m} (p_q \longrightarrow \neg u_j(i))) \vee (\neg u_j(i-1) \longrightarrow \neg u_j(i))))$$

$$(d) \bigwedge_{i=0}^{p-1} \bigwedge_{j_1=i.m+1}^{(i+1).m} \bigwedge_{q=1}^{|U|} ((p_{j_1} \longrightarrow \neg u_q(i+1)) \Rightarrow (\neg u_q(i+1) \wedge Dels(p_{j_1}, u_q) \wedge (\bigwedge_{j_2=i.m+1, j_2 \neq j_1}^{(i+1).m} (Adds(p_{j_2}, u_q) \Rightarrow (p_{j_2} \prec p_{j_1}))))))$$

$$(e) \bigwedge_{i=0}^{p-1} \bigwedge_{j_1=i.m+1}^{(i+1).m} \bigwedge_{q=1}^{|U|} ((p_{j_1} \longrightarrow u_q(i+1)) \Rightarrow (u_q(i+1) \wedge Adds(p_{j_1}, u_q) \wedge (\bigwedge_{j_2=i.m+1, j_2 \neq j_1}^{(i+1).m} (Dels(p_{j_2}, u_q) \Rightarrow (p_{j_2} \prec p_{j_1}))))))$$

$$10. \bigwedge_{i=0}^{p-1} \bigwedge_{j_1=(i.m+1)}^{(i+1).m} \bigwedge_{j_2=(i.m+1), j_2 \neq j_1}^{(i+1).m} \bigwedge_{j_3=(i.m+1), j_3 \neq j_2, j_3 \neq j_1}^{(i+1).m} (((p_{j_1} \prec p_{j_2}) \wedge (p_{j_2} \prec p_{j_3})) \Rightarrow (p_{j_1} \prec p_{j_3}))$$

$$\bigwedge_{i=0}^{p-1} \bigwedge_{j_1=(i.m+1)}^{(i+1).m} \bigwedge_{j_2=(i.m+1), j_2 \neq j_1}^{(i+1).m} \neg((p_{j_1} \prec p_{j_2}) \wedge (p_{j_2} \prec p_{j_1}))$$

$$\bigwedge_{i=0}^{p-1} \bigwedge_{j_1=(i.m+1)}^{(i+1).m} \neg(p_{j_1} \prec p_{j_1})$$

Figure 2: The schemas for generating the unifying encoding.

the $O(\frac{k^2}{p^2} | U |)$ causal links can be threatened by any of the $O(\frac{k}{p})$ steps in the same region as the contributor and consumer of the causal link, and there are p regions, $O(\frac{k^3}{p^2} | U |)$ clauses are required to resolve the potential threats. Stating that a step has the effects and pre-conditions of its action binding requires $O(k | O |)$ clauses. Stating explanatory frame axioms requires $O(k | U |)$ clauses. Ignoring the clauses generated by schemas that do not dominate the asymptotic size, we find that the unifying encoding contains $O(k | O |^2 + \frac{k^3}{p^2} | U | + k(| O | + | U |))$ clauses.

The maximum clause length is a constant and hence the sum of the clause lengths is the same as the number of clauses. The number of clauses and variables is thus minimum when $p = k$ and maximum when $p = 1$. Note that since the number of actions and fluents are constants for a given domain, we use the power of the variable k to compare the sizes of different encodings of a problem. Also note that the maximum arity of an action is generally greater than or equal to the maximum arity of a fluent. For example, in blocks world the maximum arity of an action is 3 (e.g. `move(A,B,C)` for moving block A from top of block B to top of block C) while the maximum arity of a fluent (e.g. `on(A,B)`) is 2. In the transportation logistics domain with planes and no trucks, the maximum arity of an action is 3 (e.g. `load(Package1,Paris,Plane1)` for loading Package1 into Plane1 at Paris) while the maximum fluent arity is 2 (e.g. `at(Package1,Paris)`). As a result, the number of ground actions is generally greater than or equal to the number of ground fluents (that is, $| O | \geq | U |$). Thus, the $k | U |$ term may be dropped from the expressions for the asymptotic number of variables and clauses.

We show next how the two important encodings from [Kautz et al. 1996] can be viewed

as instantiations of the unifying encoding. The state-space encoding with explanatory frame axioms is the smallest of all the encodings. Explanatory frame axioms explain any change of a fluent's truth value over a unit time interval by stating that some action is responsible for such change. A complete description of the world state is represented at all k time steps in a k step encoding of this type. Hence, this encoding can be interpreted as a special case of the unifying encoding for which $p = k$. This shows that the number of variables and clauses in a state-space encoding with explanatory frame axioms is $O(k(|O| + |U|))$ and $O(k(|U| + |O|) + k|O|^2)$ respectively. This is confirmed by previous findings [Kautz et al. 1996]. The causal encoding is the largest of all the current plan encodings. No description of the world state is represented at any point of time. Hence, this encoding can be interpreted as a special case of the unifying encoding where there is only one region ($p = 1$). Now we see that the number of variables and clauses in the causal encoding is $O(k(|O| + |U|) + k^2|U|)$ and $O(k(|O| + |U|) + k|O|^2 + k^3|U|)$ respectively. This also agrees with the sizes reported by [Kautz et al. 1996].

6 Other Hybrid Encodings

It should be noted that there are other ways, besides the unifying encoding, of developing hybrid encodings by combining the key ideas of state-space and partial-order planning in the SAT framework. For example, one can augment the causal encoding of [Kautz et al. 1996] with the contiguity constraint, $*$, on steps. This allows some steps in a plan to be contiguous and others to be partially ordered. One has to then also state the relation between contiguity and partial order to avoid inconsistency. That relation is

$$\bigwedge_{i=1}^k \bigwedge_{j=1, i \neq j}^k ((p_i * p_j) \Rightarrow ((\bigwedge_{q=1, q \neq i, q \neq j}^k \neg((p_i \prec p_q) \wedge (p_q \prec p_j))) \wedge (p_i \prec p_j)))$$

However, this adds $O(k^2)$ variables of type $p_i * p_j$ and $O(k^3)$ clauses to state the semantics of contiguity. Furthermore, we do not have descriptions of any states. Though this encoding has a hybrid nature, it is bigger than the causal encoding of [Kautz et al. 1996].

After proving the following result on hybrid encodings, we lay out all hybridization possibilities and, after a discussion of some more examples, show that no hybrid encoding is smaller than the state-space encoding with explanatory frame axioms.

Theorem. Any propositional encoding which has at least one pair of steps p_i and p_j such that $(p_i \prec p_j)$ and p_i, p_j belong to the same region is not minimal.

Proof - Let the encoding contain a total of k steps. Since there exists a region of the encoding such that p_i and p_j belong to this region, the number of regions, p , in the encoding must be less than k . As shown in Section 5, $p = k$ in the smallest instance of the unifying encoding. Varying the distribution of steps and varying the nature of ordering between the steps does not yield a minimal instance. Hence, the result. \square

We next examine some variants of the unifying encoding.

Two key notions in state-space planning are (1) World state and (2) Contiguity of steps. Two key notions in causal planning are (1) Partial order on the steps and (2) Causal links (or the fact that a step must rely on other steps and not on world states, to make its preconditions available). To consider all hybridization possibilities, we define an encoding to be

a hybrid if it contains at least one key notion from state-space planning and at least one key notion from causal planning. This view allows our conclusions to be applicable to all hybrid encodings.

We list the complete set of hybridization possibilities (that are soundness and completeness preserving) next. **1.** Contiguity and precedence constraints, only 1 region and no world state representation. One can augment the causal encoding [Kautz et al. 1996] with the contiguity constraint. One has to then also state the relation between contiguity and partial order to avoid inconsistency. This is explained at the beginning of section 6. **2.** More than one region. There are several variants of this listed next. **2.1** No world state representation. **2.1.1** Equal/Unequal distribution of steps among the regions. **2.1.1.1** All steps in a region are partially ordered. **2.1.1.2** All steps in a region are contiguous. **2.1.1.3** Some steps in a region are contiguous, others are partially ordered. **2.2** World state representation present. **2.2.1** Equal/Unequal distribution of steps among the regions. **2.2.1.1** All steps in a region are partially ordered. **2.2.1.2** All steps in a region are contiguous. **2.2.1.3** Some steps in a region are contiguous, others are partially ordered. The variant 2.2.1.1 with an equal distribution of steps among the regions is the unifying encoding in Figure 1.

6.1 Variants of the Unifying Encoding

In this subsection, we analyze three hybrid variants of the unifying encoding and show why they cannot be smaller than the state-space encoding with explanatory frame axioms. These serve as examples for verifying the argument in the theorem at the beginning of Section 6.

6.1.1 Removing the State Representation

Though there is no state representation in this encoding, regions still exist (as shown in Figure 3). Thus, each step in the i th region precedes each step in the j th region, where $j > i$. The pre-conditions of the i th step in the j th region may be established either by the initial state or by any of the steps in any of the 0 th through j th regions. Hence, the total number of causal link variables in the encoding is $O((m^2 + 2m^2 + 3m^2 + \dots + pm^2) | U |)$, that is $O(\frac{p(p+1)}{2}m^2 | U |)$. Since $k = (p.m)$, the total number of causal link variables, $(O(p^2m^2 | U |))$, is the same as $O(k^2 | U |)$.

A causal link between the steps p_i and p_j may be threatened by any step in the regions containing p_i and p_j . Thus, the total number of clauses required for threat resolution is $O((m^3 + 4m^3 + 9m^3 + \dots + p^2m^3) | U |)$, that is $O(\frac{p(p+1)(2p+1)}{6}m^3 | U |)$. Since $k = p.m$, the total number of threat resolution clauses, $(O(p^3m^3 | U |))$, is $O(k^3 | U |)$. Next, we consider some variations of this encoding.

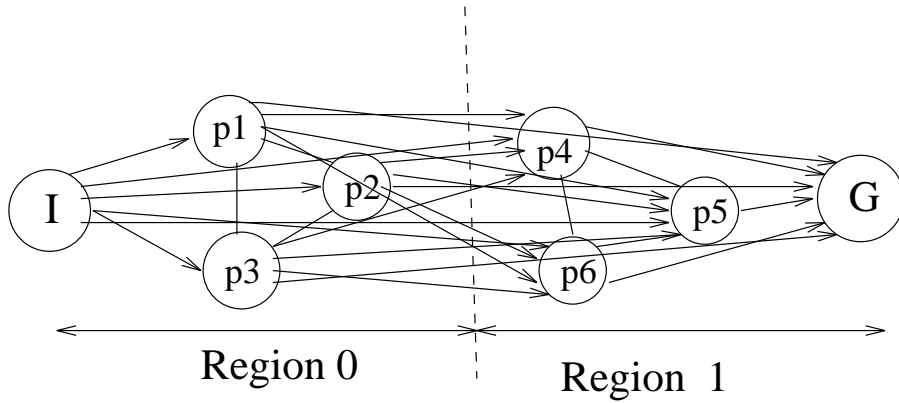


Figure 3: In the absence of state representation more causal link variables are required and, thus, more possibilities of threats exist.

6.1.2 Varying the Distribution of Steps

So far, we have assumed that the k steps in the encoding were equally distributed among the p regions. Now we consider a variant of the encoding in Section 6.1.1 where different regions may have a different number of steps. Let the i th region, $i \in [0, p - 1]$, have k_i steps so that $\sum_{i=0}^{p-1} k_i = k$. The total number of causal link variables in the encoding will then be

$$O\left(\left(\sum_{i=0}^{p-1} \sum_{j=0}^i (k_i \cdot k_j)\right) \mid U \mid\right).$$

The total number of clauses required for threat resolution will be

$$O\left(\left(\sum_{i=0}^{p-1} \sum_{j=0}^i \sum_{q=j}^i (k_i \cdot k_j \cdot k_q)\right) \mid U \mid\right).$$

In addition to these, $O(k \cdot \mid O \mid^2)$ clauses will be needed to ensure that a step is not bound to more than one action. Furthermore, an additional $O(k \cdot (\mid O \mid + \mid U \mid))$ clauses are needed to state that steps have the pre-conditions and effects of their action bindings and that steps add/need/delete a fluent only if they are bound to actions which add/need/delete that same fluent.

$$\sum_{i=0}^{p-1} k_i = k$$

and

$$\sum_{i=0}^{p-1} k_i < \left(\sum_{i=0}^{p-1} \sum_{j=0}^i (k_i \cdot k_j)\right) < \sum_{i=0}^{p-1} \sum_{j=0}^i \sum_{q=j}^i (k_i \cdot k_j \cdot k_q).$$

Now we note that this encoding is not minimal since the minimal encoding contains $O(k(\mid U \mid + \mid O \mid) + k \cdot \mid O \mid^2)$ clauses and $O(k(\mid U \mid + \mid O \mid))$ variables.

6.1.3 Two Types of Regions

So far we have considered instances of the unifying encoding where all steps in the regions of the encoding are partially ordered. We will now consider instances that can have two

types of regions - (i) regions where all steps are contiguous and (ii) regions where all steps are partially ordered. Let p_1 be the number of regions that contain contiguous steps and p_2 be the number of regions that contain partially ordered steps, so that $p_1 + p_2 = p$. Let $j_1, j_2, j_3, \dots, j_{p_1}$ be the indices of the regions that contain contiguous steps ($k_{j_1}, k_{j_2}, \dots, k_{j_{p_1}}$ being the number of steps in each of these regions) and q_1, q_2, \dots, q_{p_2} be the indices of the regions that contain partially ordered steps ($k_{q_1}, k_{q_2}, \dots, k_{q_{p_2}}$ being the number of steps in each of these regions).

Now, the pre-conditions of a step p_i in a region s can be established by the initial state, any of the steps in any of the regions 0 to $(s - 1)$, any step in the region s (if the steps in the region s are partially ordered), or any step in the region s that occurs before p_i (if the steps in region s are contiguous). We need not use causal link variables $p_i \xrightarrow{f} p_j$ if p_i and p_j both belong to a region, all steps of which are contiguous. In this case, one can state that p_j gets its pre-conditions either because some preceding steps from its region add them or because there exists a causal link, $p_k \xrightarrow{f} p_j$, from a step p_k in some previous region. We thus use causal link variables only for a contributor-consumer pair that does not belong to a region with all the steps contiguous. Hence, the total number of variables in the encoding is

$$O\left(\left(\sum_{i=j_1}^{j_{p_1}} k_i + \sum_{r=q_1}^{q_{p_2}} k_r^2\right) \cdot |U|\right).$$

Consider a region with four steps, all of which are contiguous. Let $p_i * p_j, p_j * p_a, p_a * p_b$ be the contiguity relations. To deal with any threats (caused by the steps in this region), one can state that the deleted pre-conditions must be re-established, e.g. $(Dels(p_i, u_z) \wedge Needs(p_b, u_z)) \Rightarrow (Adds(p_a, u_z) \vee Adds(p_j, u_z))$. Such threats posed by steps in the same region can be handled with $O(k_s^2 |U|)$ clauses, where k_s is the number of steps in the region

(all steps are contiguous). Hence, the total number of clauses required for threat resolution will be

$$O\left(\left(\sum_{i=j_1}^{j_{p_1}} k_i^2 + \sum_{r=q_1}^{q_{p_2}} k_r^3\right) |U|\right).$$

Since

$$\left(\sum_{i=0}^{p-1} k_i\right) < \left(\sum_{i=j_1}^{j_{p_1}} k_i + \sum_{r=q_1}^{q_{p_2}} k_r^2\right) < \left(\sum_{i=j_1}^{j_{p_1}} k_i^2 + \sum_{r=q_1}^{q_{p_2}} k_r^3\right),$$

this instance of the encoding in 6.1.1, with two types of regions, is bigger than the minimal instance.

Let us consider an instance of the encoding in 6.1.1 with regions of three types. This instance can be derived from the instance which has two types of regions by changing the nature of ordering in some regions so that some steps in these regions are contiguous and others are partially ordered. However, for these new regions, we will need both $p_i * p_j$ type contiguity variables and the $p_i \prec p_j$ type precedence variables. We will also need additional clauses to relate the contiguity and partial order. Thus, the instance with three types of regions is bigger than the instance with two types of regions and, therefore, isn't minimal.

The arguments for the three encodings in Sections 6.1.2 and 6.1.3 can be also applied to the three corresponding instances of the unifying encoding which do have the notion of state. Thus, it can be shown that they are not minimal. These three encodings are - (a) Equal distribution of steps with regions of one type (all steps in a region being partially ordered), (Section 6.1.2), (b) Unequal distribution of steps, with regions of the same type as in (a) (Section 6.1.3), and (c) Unequal distribution of steps and two types of regions (Section 6.1.3). Next, we explain why the size-related arguments for the 3 encodings in 6.1.2 and 6.1.3 hold for corresponding instances of the unifying encoding.

Version (a) of the unifying encoding is the unifying encoding itself, which is not minimal if $1 \leq p < k$. If the steps are not equally distributed among the regions, there is some region that has more than $\frac{k}{p}$ steps. Let this region have k' steps. If these steps are partially ordered, there will be $O(k'^2 | U |)$ causal link variables for this region alone. If these steps are contiguous (e.g. $p_{j1} * p_{j2}, p_{j2} * p_{j3} \dots$), there still will be $O(k'^2 | U |)$ causal links for the region. This shows that type (b) and type (c) instances of the unifying encoding cannot be minimal.

Having fewer regions than the number of plan steps requires more clauses and variables to represent the planning constraints. Removing the representation of states increases the size still further since all interactions between the steps of a region i and the steps in the regions preceding i need to be represented. If steps are not equally distributed among the regions, the size of the encoding worsens (increases).

This analysis leads us to the following result which is just another way of stating the Theorem near the beginning of this section.

Theorem. No hybrid encoding has fewer variables, fewer clauses, or a lower sum of clause lengths than the smallest encoding (state-space encoding with explanatory frame axioms) in non-incremental domain-independent planning.

7 Incremental Planning

Current implementations of planning as satisfiability have been applied to non-incremental planning where each new problem is solved from scratch by bounding the number of plan

steps. These implementations with the exception of [Mali 1999(b)], have not explored the potential of the satisfiability paradigm for incremental planning. In incremental planning new steps can be added, or existing steps removed, from an existing plan. It should be noted that the incremental planning we deal with here is not exactly the same as the notion of plan reuse (where an attempt is made to recycle as much of an old plan as possible). Instead, we represent the old plan in an encoding leaving the possibility of removal of the steps from the old plan open. This is not a disadvantage since conventional strategies of plan reuse [Kambhampati & Hendler 1992] are only heuristic and do not guarantee maximum plan recycling.

We are interested in examining whether or not the unifying encoding provides any advantages in incremental planning. This line of investigation is inspired by the intuition that the unifying encodings may give us the best of worlds - lower size of state-space encodings and the flexibility offered by causal encodings in allowing reordering of steps. Flexibility in reordering steps is very important in incremental planning. To allow the addition of new actions to an existing plan, we must increase the number of steps in an encoding. Thus, the encoding for the incremental planning scenario has three parts - (i) a part that represents the constraints from the old plan, (ii) a part that represents the constraints between the new steps, and (iii) a part that represents the interactions between the old plan's steps and the new steps.

Just as the number of plan steps is bounded by k in non-incremental planning, we bound the number of plan steps in the incremental planning as well (by $(k + q')$). Here, q' depends on the specific way of generating an encoding for incremental planning. k is number of steps

in old plan and q' is the number of new steps included in an encoding to allow inclusion of more actions. We do not consider the criterion of maximally recycling the old plan. This criterion serves as only a restriction on the models of the propositional encodings that we synthesize next, leaving their applicability unaffected. Maximal recycling of a plan can be achieved in the following three ways - (i) by converting a SAT encoding into 0-1 ILP encoding and choosing minimization of the sum of boolean variables for addition of new actions as the objective function, (ii) by adding several highly weighted unit clauses of the form $\neg o_i(t)$ to a SAT encoding, where t is either a new time step or o_i does not occur in the old plan at time t and then using a weighted MAX-SAT solver to maximize the sum of weights of satisfied clauses, or (iii) by solving a series of encodings generated with $q' = 1, 2, 3, \dots$ until a plan is found for the new planning problem, allowing only one action to occur at each of the q' steps.

To allow the addition of new actions to a k step plan, we include q' new steps in an encoding so that q new actions can be added where $q' \geq q$. Some state-space encodings allow parallel actions and in these cases more than q actions may be added. The relationship between q and q' is encoding dependent and will become clear in section 7.1. Note that when comparing the sizes of encodings for incremental planning, we consider the powers of both k and q' in the asymptotic expressions. Actions from an old plan can be removed by changing the truth assignments of appropriate variables, e.g. an occurrence of o_j can be removed by making $(p_i = o_j)$ false and $(p_i = \phi)$ true. Next, we discuss the key modifications that various current encodings need to handle incremental planning.

We first show how state-space encodings can be modified to handle incremental planning.

We then compute the sizes of the modified state-space encodings and compare it with the size of the causal encoding and a particular instance of the unifying encoding. We also examine the likelihood of the presence of non-minimality in the plans found by solving each of the encodings in sections 7.1.1, 7.1.2, 7.2 and 7.3. A non-minimal plan is a plan from which some actions can be removed without affecting its correctness. As argued in [Bacchus & Kabanza 2000], [Kambhampati 1995], and [Smith & Peot 1996], non-minimal plans are likely to be generated in domains where the effects of actions can be undone and re-established an arbitrarily large number of times. For example, loading and unloading packages into and from a truck, painting and scratching a wall, stacking and unstacking blocks, and flying planes or driving vehicles to and fro could all lead to non-minimal plans. Most of the domains currently used to test planners do have such actions. Since significant cost of execution is associated with such non-minimality in practice, it is important to control extra actions. It should be noted that there are three reasons behind non-minimality in the plans found in the satisfiability framework - (i) Choice of an incorrect k (if too many steps are chosen, redundant actions may be present in the plan leading to non-minimality), (ii) choice of the satisfiability solver, and (iii) Choice of the encoding. In this section and section 9, we investigate the role of choice of an encoding in generation of non-minimal plans. Different types of encodings (with the same or different number of steps) for the same problem may lead to different degrees of non-minimality in the plans found. For example, if all minimal plans for a planning problem have k actions, a k step encoding that allows parallel actions (that is, multiple actions to occur at a time step) may yield non-minimal plans (having more than k actions). On the other hand, a k step encoding that allows only one action to occur

at a time step will yield a minimal plan.

7.1 Adapting the State-space Encodings

We next discuss two schemes for adapting the state-space encoding to the generalized-incremental planning scenario.

7.1.1 Scheme 1

Let us consider the two step plan in Figure 4 where the steps are mapped to the actions o_1, o_2 respectively and are contiguous. Let us assume that a maximum of 3 actions will have to be added to this plan at no more than 3 new time steps to solve a new planning problem. These actions may be added at arbitrary places, disturbing the existing order of actions in the old plan. The actions from the old plan may be removed. Hence, we not only need to have the option of mapping the old steps to no-ops (thus removing the actions occurring at these steps), but we may also have to make multiple copies of the old plan to allow the possible interleaving of new and the old actions. For this reason we reserve multiple locations for any new actions. This modification allows state-space encodings to simulate the ability of causal encodings to accommodate new steps. In particular, if we have k steps in the old plan we make k copies of the old plan. This requires k^2 steps. Also, if we want to add q new actions to the old plan, we reserve $(k + 1)$ blocks (at different places) for these new actions, each block being capable of accommodating q actions. Hence this requires $(k + 1).q$ new steps.

Hence the state-space encoding modified for incremental planning where actions may be added at a maximum of q new time steps and any number of old actions can be removed requires $(k^2 + (k + 1)q)$ steps. In this scheme, $q' = (k^2 + (k + 1).q - k)$.

Note that we are interested in synthesizing new plans with a total of $(k + q)$ or fewer

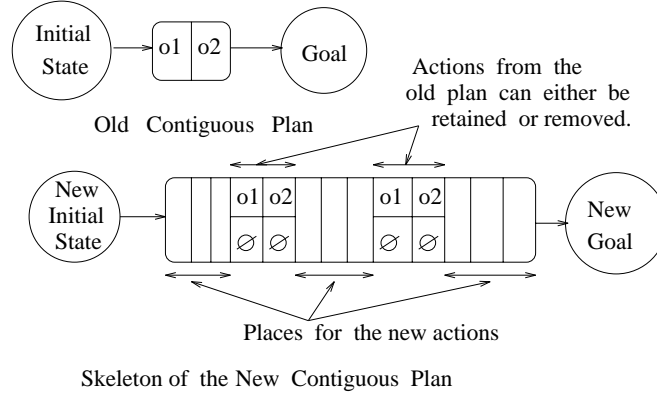


Figure 4: Scheme 1 - Incremental planning with a contiguous plan by making multiple copies of the old plan and including multiple blocks of steps for new actions ($k = 2, q = 3$).

non-null actions, occurring at $(k + q)$ or fewer steps. Let us find the size of the modified state-space encoding. Since the truth of the action occurrences and the truth of the precondition and effect propositions is represented at each time step, and since each of the new steps can be mapped to any of the actions from the domain, the modified state-space encoding for incremental planning will have $O((k^2 + (k + 1)q) | U | + (k + 1)q | O |)$ variables and $O((k^2 + (k + 1)q) | U | + (k + 1)q | O | + (k + 1)q | O |^2)$ clauses, due to the explanatory frame axioms and the possibility of the occurrence of any action from the set O at any of the $(k + 1)q$ new time steps. Terms like $(k + 1).q. | O |$ that do not contain higher powers of k or q are not added to the expressions shown in Figure 5. The size of the domain of the binding of each copy of each step from the old plan is 2 rather than $| O |$.

Though the size of a state-space encoding for non-incremental planning is linear in terms of the number of steps, this is no longer the case for incremental planning based on this scheme as we showed here. Also, since we need a plan of only $(k + q)$ steps, the extra $(k^2 + k.q - k)$ steps must be mapped to no-ops (to reduce the potential for non-minimality

in the plans generated). Though k copies of the old plan are made (and thus k copies of each occurrence of each of the plan's actions are made), no more than one copy of each occurrence of each action should be used. If more occurrences of these actions are required, the action can still occur at the new $(k + 1).q$ steps introduced to allow new actions. Thus, the following clauses which capture this for controlling non-minimality are soundness and completeness preserving. Here, o_i denotes i th action from the old plan.

$$\bigwedge_{q'=1}^k \bigwedge_{i=1}^k (o_i(q'.q + (q' - 1).k + i - 1)) \Rightarrow \left(\bigwedge_{q_1=1, q_1 \neq q'}^k \neg o_i(q_1.q + (q_1 - 1).k + i - 1) \right)$$

This specification requires $O(k^3)$ clauses.

Also, we need to control non-minimality that may arise out of redundant new actions occurring in the multiple blocks of steps reserved for their inclusion. Thus, if a non-null action occurs at one of the $(k + 1)$ steps reserved for it, the null action must occur at the remaining k steps. This restriction applies to each of the actions that may occur at any of the $(k + 1).q$ steps reserved for the introduction of new actions.

$$\bigwedge_{q'=1}^{(k+1)} \bigwedge_{i=1}^q (\neg \phi((q' - 1).q + (q' - 1).k + i - 1)) \Rightarrow \left(\bigwedge_{q_1=1, q_1 \neq q'}^{(k+1)} \phi((q_1 - 1).q + (q_1 - 1).k + i - 1) \right)$$

This requires $O(k^2.q)$ additional clauses. Thus, the total number of clauses in an encoding augmented with these non-minimality controlling clauses will be $O((k^2 + (k + 1).q). |U| + k^3 + k^2.q + (k + 1).q. |O| + (k + 1).q. |O|^2)$.

7.1.2 Scheme 2

Instead of making multiple copies of an old contiguous plan and reserving multiple places for adding new actions, in this scheme we specify that each action o_i from the old plan will either occur at any of the $[0, k + q - 1]$ time steps or will never occur at all (for this case, we

create a special variable ϕ'_i). This is stated as $((\bigvee_{j=0}^{k+q-1} o_i(j)) \vee \phi'_i)$. Thus, if no occurrence of o_i is required, the preceding clause can still be satisfied by setting ϕ'_i to true. New actions may occur at any time step.

This $(k+q)$ step state-space encoding with explanatory frame axioms will contain $O((k+q).(|U| + |O|) + (k+q).|O|^2)$ clauses and $O((k+q).(|U| + |O|))$ variables. In this scheme, $q' = q$. Thus, the size of this state-space encoding is the same as the size of a $(k+q)$ step state-space encoding for non-incremental planning. If an action occurs in the old plan once, then the action should occur in the $(k+q)$ step plan at no more than $(q+1)$ times (since q new steps are added). But, the $((\bigvee_{j=0}^{k+q-1} o_i(j)) \vee \phi'_i)$ style of specification allows an action to occur more number of times. In general, let us say that an action o_i occurs s times in the old plan of k steps, where $s < k$. Even if all q new steps are mapped to o_i , the total number of occurrences of o_i in the final plan of $(k+q)$ steps should not exceed $(q+s)$. To state this in the encoding we will require $O((k+q)^{(q+s+1)})$ clauses. This large number of clauses will have to be stated for each action in the old plan. If these clauses are not included, we will have an encoding whose models may be non-minimal plans which users (who want the old plans to be handled in a certain way) may not approve of. Similarly, a new action should not occur more than q times in the final plan of $(k+q)$ steps. Control over the number of occurrences of new actions is not guaranteed by the current state-space encodings with explanatory axioms. To enforce this restriction, one will have to add $O((k+q)^{(q+1)})$ new clauses for each action that does not occur in the old plan, but may occur in the final plan.

Another solution to curb non-minimality is to add mutual exclusivity of actions restrictions to the encoding. However, if one does this, one still does not have control over the

number of occurrences of a particular action. This restriction only makes sure that an encoding containing $(k + q)$ steps will yield a plan with no more than $(z + q)$ non-null actions (z being the number of actions in the old plan).

7.2 Using the Causal Encoding

If we are to use the causal encoding for incremental planning, we do not need redundant steps as in the state-space encoding based on the scheme 1 in Section 7.1.1. The reason is that since causal encodings impose a partial order on the steps, they are directly compatible with incremental planning. Thus, if no more than q actions are to be added to the old plan, only q new steps are needed. The number of clauses and variables in the causal encoding for incremental planning will then be $O((k + q)^3 \cdot |U| + q \cdot |O|^2 + q \cdot (|O| + |U|))$ and $O((k + q)^2 \cdot |U| + q \cdot (|O| + |U|))$ respectively. The term $(q \cdot |O| + q \cdot |U|)$ is not added to the number of variables in Figure 5, because it does not contain higher powers of q . Likewise, the term $(q \cdot |O|^2 + q \cdot (|O| + |U|))$ is not added to the number of clauses in Figure 5 since it does not contain higher powers of q .

Causal encodings do not have the problem of the number of occurrences of an action exceeding limits. A step in a causal encoding is not bound to more than one action. If an action o_i occurs s times in an old plan, exactly s occurrences of this action will be represented for the old plan in the causal encoding for the incremental planning scenario. Like before, an action from an old plan is removed by mapping the corresponding step to the no-op. For example, to remove the action o_j from an old plan during the incremental planning process, some mapping $(p_i = o_j)$ is changed to $(p_i = \phi)$. Furthermore, since a step from an old plan can be either retained or removed and cannot be mapped to some other non-null action, the

number of occurrences of newly introduced actions is controlled as well. We next investigate the potential of the unifying encoding for handling incremental planning.

7.3 Using the Unifying Encoding

Since the q new steps may be arbitrarily distributed among the existing p regions of the encoding, we need to include extra steps to account for this distribution. Hence, when q new steps are to be added to an existing plan we add $q.p$ new steps to the existing encoding, allocating q steps to each of the p regions. Though this allows a flexible addition of new steps, the possibility of generating highly non-minimal plans is also left open. To make sure that no more than q non-null actions are added to the old plan, we can add more constraints to the encoding. However the number of such constraints will be prohibitively large ($O(p^{(q+1)}.q^{(q+1)})$). These constraints consider all choices of q out of $q.p$ steps and specify that if any of these q steps is not mapped to the no-op, all other $(p - 1).q$ steps must be mapped to no-ops.

Since each region in the encoding now has q more steps, each region will have $O((\frac{k}{p} + q)^2 . | U |)$ causal links. Since there are p regions in the encoding, there will be $O((\frac{k}{p} + q)^2 . p . | U |)$ causal link variables and $O((\frac{k}{p} + q)^3 . p . | U |)$ clauses (for resolving any threats). The term $(q.p.(| O | + | U |))$ is not added to the number of variables in the unifying encoding, in Figure 5, since it does not contain higher powers of q . This term represents the total number of new variables of type $p_i = o_j, Adds(p_i, f), Needs(p_i, f),$ and $Dels(p_i, f),$ that are needed due to the addition of $q.p$ new steps. Likewise, the term $(q.p.(| O | + | U |) + q.p. | O |^2)$ is not added to the number of clauses in the unifying encoding, in Figure 5, since it does not contain higher powers of q . This term represents the total number of new clauses required

to state that new steps have the effects and pre-conditions of their action bindings and that the new steps cannot be bound to more than one action.

Since the number of variables and clauses in the unifying encoding for the incremental planning scenario is expressed as a function of p , it is possible to find the value of p for which the asymptotic size of the encoding will be minimized.

Since the number of variables and clauses are functions only of the number of regions in the encoding (we denote these functions by $v(p), c(p)$ respectively) when k and q are kept constant, we can differentiate $c(p)$ and $v(p)$ to get the value of p for which the number of clauses or variables is minimized. Then, we can also get the minimum number of variables and clauses.

Solving

$$\frac{d(v(p))}{dp} = 0$$

where

$$v(p) = O\left(\left(\frac{k}{p} + q\right)^2 \cdot p \cdot |U|\right)$$

we get $p = \frac{k}{q}$. Thus, the minimum number of variables is the value of $v(p)$ at $p = \frac{k}{q}$.

Therefore, the minimum number of variables is $O(k \cdot q \cdot |U|)$.

$$v''(p) = \frac{d^2(v(p))}{dp^2} = \frac{2 \cdot k^2}{p^3}$$

and $v''\left(\frac{k}{q}\right) > 0$, confirming that a minimum indeed exists at $p = \frac{k}{q}$.

Solving

$$\frac{d(c(p))}{dp} = 0$$

where

$$c(p) = O\left(\left(\frac{k}{p} + q\right)^3 \cdot p \cdot |U|\right)$$

we get

$p = \frac{2k}{q}$. Furthermore,

$$c''(p) = \frac{d^2(c(p))}{dp^2} = \frac{6 \cdot k^3}{p^4} + \frac{6 \cdot k^2 \cdot q}{p^3}$$

Thus, $c''\left(\frac{2k}{q}\right) > 0$ which confirms that $p = \frac{2k}{q}$ is a minimum. Hence, the number of variables and clauses is minimum for different values of p . Since the difference between the number of clauses for $p = \frac{k}{q}$ (where the number of variables is minimum) and $p = \frac{2k}{q}$ is negligible and the number of clauses for both of these values of p is $O(k \cdot q^2 \cdot |U|)$, we choose $p = \frac{k}{q}$. Given that $1 \leq p \leq k$, $p = \frac{k}{q}$, $q \geq 1$, and $q \leq k$, there is a limit on the number of new actions that can be added. The term $k \cdot (|O| + |U|)$ is not added to the number of variables in the unifying encoding with $p = \frac{k}{q}$ in Figure 5 since it does not contain higher powers of q or k . Likewise, the term $(k \cdot (|O| + |U|) + k \cdot |O|^2)$ is not added to the number of clauses in the unifying encoding with $p = \frac{k}{q}$ in Figure 5 since it too does not contain higher powers of q or k .

A comparison of the sizes of the various encodings (without the extra clauses for controlling potential non-minimality) we have considered is given in Figure 5. Since the steps in an old plan remain in the same region and are not reordered by moving them from one

region to another, can the unifying encoding have models (plans) of $(k + q)$ steps that can be found only by reordering the steps from the old plan? Since each of the new $q.p$ steps can be mapped to any of the $|O|$ actions in the domain, such plans of $(k + q)$ steps can be found when $q.p \geq (k + q)$. For this, $p \geq (\frac{k}{q} + 1)$ is needed which is different from the optimum case $p = \frac{k}{q}$.

<i>Encoding</i>	<i># Variables</i>	<i># Clauses</i>
<i>State-space (Sch. 1)</i>	$O((k^2 + (k + 1).q). U)$	$O((k^2 + (k + 1).q). U)$
<i>Causal</i>	$O((k + q)^2. U)$	$O((k + q)^3. U)$
<i>Unifying</i>	$O((\frac{k}{p} + q)^2.p. U)$	$O((\frac{k}{p} + q)^3.p. U)$
<i>Unifying ($p = \frac{k}{q}$)</i>	$O(k.q. U)$	$O(k.q^2. U)$
<i>State-space (Sch. 2)</i>	$O((k + q).(U + O))$	$O((k + q).(U + O))$

Figure 5: A comparison of the sizes of the encodings for incremental planning. A plan of k steps is included in the encodings to solve a new problem in an incremental style. Upto q new actions may be added and old actions may be removed.

Though the aim of adapting the encodings to generalized-incremental planning was to see if an adapted unifying encoding is smaller than an adapted state-space and an adapted causal encoding, we also offer comments on encodings of generalized-incremental and non-incremental planning. If the number of new steps added (q') is very low in comparison with k , the adapted causal and unifying encodings will generally be smaller than the $(k + q')$ step naive causal and naive unifying encodings, respectively. This is because of a significant amount of simplification that can be carried out as discussed next. Consider a causal link variable $p_i \xrightarrow{f} p_j$ where p_i is a step in an old plan and p_j is a new step. Let o_x be the action

binding of p_i . This causal link variable should be included in an encoding for incremental planning only if f is contained in the list of add effects of o_x . Otherwise, this variable need not be used in the encoding. Similarly, let us consider the causal link variable $p_a \xrightarrow{f} p_b$ where p_a and p_b are steps from an old plan. Let o_x and o_y be the action bindings of p_a and p_b . Again, the causal link variable should not be included in an encoding if neither the list of add effects of o_x nor the list of pre-conditions of o_y contains f . Thus, using the known action bindings of steps from an old plan, one can significantly reduce the number of variables and clauses in the causal and unifying encodings for incremental planning. As a result, the causal encoding for incremental planning will always be smaller than naive causal encoding and the unifying encoding for incremental planning will be smaller than naive unifying encoding if $q' \lll k$.

7.4 Plan Completion

A particular type of incremental planning that [Milani et al. 1998] mention is that of plan completion. In plan completion, an incomplete plan is extended to solve a problem without removing actions or modifying any constraints (like the precedence relations between actions) from the old plan. This can be considered as a special case of plan reuse where the entire plan is reused. In this subsection, we extend the SAT paradigm to plan completion. To prevent a violation of the constraints from the incomplete plan, a conjunction of the constraints is included in the encoding. A contiguous plan of k steps which is to be completed by adding q actions at arbitrary places is shown in Figure 6. It can be completed by solving an encoding in which $(k + 1)$ blocks, each long enough to accommodate q actions, are included. Thus, the encoding for completing a k step contiguous plan contains $O((k + 1).q.(|U| + |O|))$

variables and $O((k + 1).q.(|U| + |O|) + (k + 1).q. |O|^2)$ clauses.

Now, let us consider completing a partially ordered plan of k steps. The particular instance of the unifying encoding where $p = \frac{k}{q}$ when used for plan completion, contains $O(k.q. |U|)$ variables and $O(k.q^2. |U|)$ clauses. A comparison of sizes of various encodings for plan completion is shown in Figure 7. From it, we see that this particular instance of the unifying encoding has fewer asymptotic variables than the state-space encoding with explanatory frame axioms for the plan completion problem. As before, terms that do not contain higher powers of q or k are not added to the expressions in Figure 7. Recall that such terms are also neglected from Figure 5.

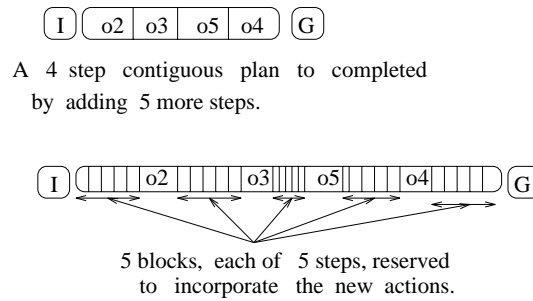


Figure 6: Completion of a contiguous plan.

<i>Encoding</i>	<i># Variables</i>	<i># Clauses</i>
<i>State-space</i>	$O((k + 1).q.(U + O))$	$O((k + 1).q.(U + O))$
<i>Causal</i>	$O((k + q)^2. U)$	$O((k + q)^3. U)$
<i>Unifying</i> ($p = \frac{k}{q}$)	$O(k.q. U)$	$O(k.q^2. U)$

Figure 7: A comparison of the encodings for the plan completion problem.

<i>Encoding</i>	<i># Variables</i>	<i># Clauses</i>
<i>State-space</i>	$O((k+q) \cdot (U + O))$	$O((k+q) \cdot U + (k+q)^{(q+s+1)})$
<i>Causal</i>	$O((k+q)^2 \cdot U)$	$O((k+q)^3 \cdot U)$
<i>Unifying</i> ($p = \frac{k}{q}$)	$O(k \cdot q \cdot U)$	$O(k \cdot q^2 \cdot U + (k+p \cdot q)^{(q+s+1)})$

Figure 8: A comparison of the encodings for the incremental planning scenario with a restriction on the number of occurrences of an action that occurs s times in an old plan. The size of the state-space encoding in this table is based on the assumption that the encoding is generated by scheme 2 of Section 7.1.2. If the state-space encoding based on scheme 1 of Section 7.1.1 is used, $O((k^2 + (k+1) \cdot q) \cdot |U| + (k+1) \cdot q \cdot |O|)$ variables and $O((k^2 + (k+1) \cdot q) \cdot |U| + k^3 + k^2 \cdot q + (k+1) \cdot q \cdot |O|^2)$ clauses are needed. As shown in section 9, another way to state the restriction about the number of occurrences of an action reduces the number of clauses in the unifying encoding with, $p = \frac{k}{q}$, from $O((k+p \cdot q)^{(q+s+1)})$ to $O(\frac{k^2}{q})$.

8 Empirical Evaluation

Our theoretical work on the sizes of hybrid encodings shows that no hybrid encoding is smallest in non-incremental domain-independent planning. Since larger encodings are not always harder to solve, we conducted an empirical evaluation of hybrid encodings. Our empirical evaluation of various instances of the unifying encoding on several benchmark domains is shown in Figure 9. The experiments were carried out on Dual Intel Pentium II 400 MHz SunOS 5.7 machine. The number of steps in the encodings were the same as the number of actions in the plans. We found that simplifying the encodings by unit propagation did not have an appreciable impact on the sizes and solving times. For example, by propagating the unit clause α , $(\alpha \wedge (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \gamma))$ can be simplified to $(\alpha \wedge \beta \wedge \gamma)$.

Our expressions for the asymptotic sizes of the instances of the unifying encoding indicate

that the ratios of the number of variables to clauses in the encodings of a problem with p_1 and p_2 regions are close to $\frac{p_2}{p_1}$ and $(\frac{p_2}{p_1})^2$, respectively. This is not the case as shown by our results in Figure 9, because of the contributions of terms that do not dominate the asymptotic size. Our implementation did contain the three types of causal link variables $u_i(t) \rightarrow u_i(t + 1)$, $u_q(t) \rightarrow p_i$, and $p_i \rightarrow u_q(t)$, in the state-space and causal encodings, where these links are not needed. The results clearly show that the hybrid encodings were neither smaller nor faster to solve than the smallest encoding. Many times domain-specific or problem-specific constraints can be propagated to reduce the size of an encoding which may make it faster to solve. In [Mali & Kambhampati 1999], it has been shown that causal encodings are strictly larger and harder to solve than state-space encodings. The sizes of different encodings for domain-independent planning differ because of the different ways of declaratively specifying the constraints (clauses) that will be true if and only if there is a plan of certain length. Thus a larger encoding does not contain more information that can be propagated to simplify it.

9 Controlling Non-Minimality of Plans

In this section, we show how various encodings can be augmented with constraints to control non-minimality of plans found by solving them. Note that minimal plans may not be same as shortest plans.

As shown in Section 7.1, one has to explicitly add extra clauses to the state-space encodings to control the number of occurrences of an action. How do the encoding sizes change when this bounded occurrences criterion is enforced? We show a comparison of the sizes in Figure 8. If an action occurs s times in an old plan, and q new steps are added (in the incremental planning scenario), the action should not occur more than $(q + s)$ times in

Domain (Steps)	$p = 1$	$p = k/6$	$p = k/4$	$p = k/3$	$p = k/2$	$p = k$
	V, C, T	V, C, T	V, C, T	V, C, T	V, C, T	V, C, T
Blocks (12)	6111	3918	3230	2916	2662	2648
	59703	19857	11975	9093	6967	5867
	16.6	46.98	1.24	1.03	0.45	0.26
Ferry (12)	5598	3605	2983	2699	2469	2455
	53670	18913	12083	9592	7762	6836
	16.95	50.72	0.83	1.15	0.25	0.2
Ferry (18)	6499	3074	-	2294	2094	2074
	99055	16134	-	7605	5924	5049
	*	2.42	-	3.49	0.41	0.28
Ferry (8)	1003	-	690	-	562	552
	6123	-	2408	-	1413	1205
	0.64	-	0.13	-	0.02	0.02
Tsp(12)	5149	3303	2725	2461	2247	2233
	49321	16426	9942	7575	5833	4941
	8.44	5.04	0.52	0.27	0.13	0.1
Tsp(8)	1801	-	1243	-	1019	1009
	10817	-	4208	-	2431	2061
	0.74	-	0.16	-	0.04	0.04
Log(12)	3790	2439	2015	1821	1663	1649
	36574	12380	7614	5876	4598	3942
	27.57	54.69	0.64	0.69	0.27	0.24
Log(6)	696	696	-	508	460	452
	3066	3066	-	1396	1068	914
	0.14	0.14	-	0.05	0.02	0.02
Tireworld(12)	5127	3282	2704	2440	2226	2212
	49431	16424	9914	7537	5787	4889
	10.57	6.02	0.68	0.61	0.28	0.21

Figure 9: Empirical results on various (hybrid) instances of the unifying encoding from Figure 1. $p = 1$ corresponds to the purely causal encoding and $p = k$ corresponds to the purely state-space encoding. V, C, T denote the number of variables, clauses and time needed to solve the encodings respectively. Times are in CPU seconds. A “*” indicates that the encoding was not solved within 5 minutes of CPU time. A “-” denotes that the encoding was not generated since the number of regions was not an integer. The descriptions of the benchmark domains used and the “satz” systematic SAT solver used are available at <http://www.cs.yale.edu/HTML/YALE/CS/HyPlans/~mcdermott.html> and <http://aida.intellektik.informatik.th-darmstadt.de/~hoos/SATLIB> respectively. Log is the transportation logistics domain and Tsp is the traveling salesperson domain.

the final plan. This is the restriction we impose on the models of the encodings. In the state-space encoding based on scheme 2 (in section 7.1.2), we have to state that $(q + s + 1)$ occurrences of an action are not allowed. It is clear that the state-space encoding with explanatory frame axioms is not the smallest because of the large number of such extra clauses required (irrespective of the scheme used to generate it). The memory required to store the unifying encoding is impractical as shown by the number of clauses in Figure 8 if we state that $(q + s + 1)$ occurrences of an action are not allowed. To reduce this, we use the following reasoning. We require that $(p - 1).q$ out of the newly added $p.q$ steps must be bound to no-ops, since only q or fewer new actions are needed. Thus, we can say that if the $(\frac{k}{p} + i)$ th step ($i \in [1, q]$) in j th region of the unifying encoding ($j \in [0, p - 1]$) is not bound to a no-op, then the $(\frac{k}{p} + i)$ th step in every other region $j' \neq j, j' \in [0, p - 1]$, should be bound to a no-op. This can be specified by the following $O(q.p^2)$ clauses.

$$\bigwedge_{j=0}^{p-1} \bigwedge_{i=1}^q (\neg(p_{((j+1).(\frac{k}{p})) + j.q+i} = \phi) \Rightarrow (\bigwedge_{j'=0, j' \neq j}^{p-1} (p_{(j'+1).(\frac{k}{p} + j'.q+i)} = \phi)))$$

If $p = \frac{k}{q}$, the number of clauses can be reduced to $O(\frac{k^2}{q})$. State-space encodings with explanatory frame axioms allow non-conflicting actions to occur in parallel. One can restrict them with $O(k \cdot |O|^2)$ clauses to ensure that only one action occurs at a time. However, this restriction still does not control the number of occurrences of individual actions. Causal encodings do not have this size explosion problem.

10 Discussion

To answer the challenge [Kambhampati 1997], we developed several hybrid encodings and found their sizes in both the domain-independent generalized-incremental and the non-

incremental planning scenarios. We showed that these encodings combining the important notions from state-space planning and partial-order planning are not smallest. We did not report the sum of the clause lengths, since these can be derived by multiplying the number of clauses by the maximum clause lengths. [Kambhampati & Srivastava 1996] have reported a universal classical planner that unifies the state-space and causal approaches to refinement planning. Our unifying encoding can be viewed as doing such a unification in the SAT planning paradigm.

It has been shown in [Mali 1999(a)] and [Mali 1999(b)], that in the presence of causal domain-specific knowledge (like the causal links and precedences in the task reduction schemas in hierarchical planning [Erol et al. 1994], [Kambhampati et al. 1998]) or in the cases of plan merging or plan reuse (where steps may have to be reordered to allow new steps to be incorporated), domain-specific knowledge or the constraints from the plans being merged or reused can be propagated to significantly simplify causal encodings. This makes them faster to solve than state-space encodings. Do these results transfer to hybrid encodings? We examine this next.

The causal encoding offers more freedom in step reordering than the hybrid encodings in incremental planning. The hybrid encodings either contain world states or contiguity on some steps, both of which prohibit certain types of reordering of steps. All steps before a world state clearly precede all steps after the world state. This limits reordering options, opening up possibilities of incompleteness. To prevent this, one has to add several extra steps to the encoding. However, adding extra steps raises the concern that generated plans may not only be non-minimal, but also may not show conformance to user intent. To prevent these

problems one has to either include a large number of constraints in the hybrid encodings, or post-process the plans to remove these problems, iteratively generating and solving the encodings until the desired plan is obtained. However, hybrid encodings may be useful in specific cases where it is guaranteed that certain step reorderings are not going to be required for solving the problems. We discuss below several kinds of planning problems where hybrid instances of the unifying encoding are highly likely to be smaller (and hopefully easier to solve) than the state-space encoding or have size closer to its size.

10.1 Large number of mutually exclusive actions

To understand how the unifying encoding can be smaller than state-space encoding in this scenario, it is necessary to know what a “planning graph” [Blum & Furst 1995] is. A planning graph is grown in the forward direction, starting with the initial state. It consists of action levels and proposition levels that alternate. A proposition level is a set of propositions. Initial state is 0th proposition/fluents level. An action level is a set of actions. The 0th action level is the set of actions whose pre-conditions appear in 0th proposition level. The i th proposition level is a union of the $(i - 1)$ th proposition level and the effects of the actions in the $(i - 1)$ th action level. Any action that occurs in a planning graph in action level i occurs in the planning graph at all action levels j where $j > i$. Similarly, any proposition that occurs in a proposition level i will occur in all j th proposition levels where $j > i$. Note that an action level also contains no-ops, which preserve the truth of a proposition if no action changing that truth occurs. Also, note that a proposition level is not the same as a world state.

A planning graph is shown in the upper part of Figure 10. Pre-conditions of actions o1,

o_2, o_3, o_4 and o_5 are shown below boxes and the effects of these actions are shown above the boxes. The 0 th proposition level is $\{a, b, c\}$ and the 0 th action level is $\{o_1, o_2, \phi_a, \phi_b, \phi_c\}$, where ϕ_a, ϕ_b , and ϕ_c are no-ops for the persistence of a, b and c .

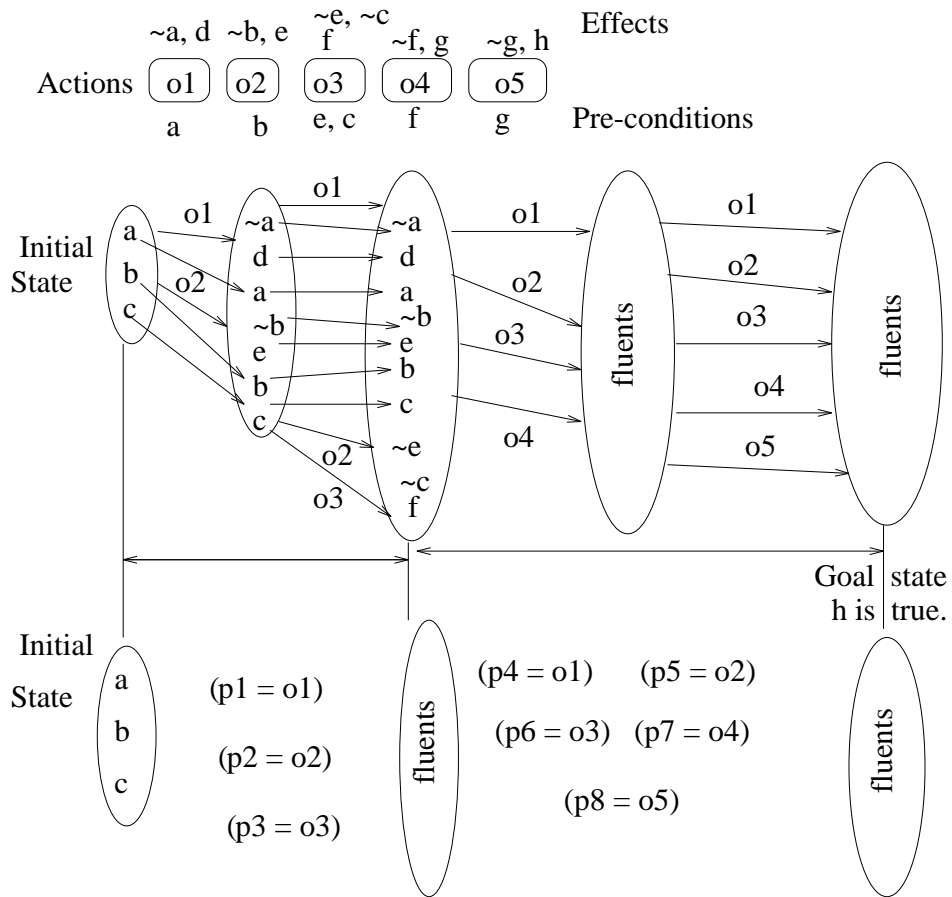
Besides constructing the planning graph, Graphplan [Blum & Furst 1995] also computes mutual exclusion relations between actions and mutual exclusion relations between propositions. The mutual exclusion relations (mutexes) are classified into static and dynamic mutexes. Static mutexes can be found by inspecting the pre-conditions and effects of actions. For example, two actions o_x and o_y are static mutex if (i) o_x deletes a pre-condition of o_y , or (ii) o_x and o_y have conflicting effects (e.g. p and $\neg p$), or (iii) o_x has a pre-condition r and o_y has a pre-condition s , such that the propositions r and s are mutex. Dynamic mutexes are found by propagating static mutexes. Static mutexes are permanent mutexes. For example, consider actions o_1 and o_2 which are static mutex at an action level i . Also, suppose that nobody other than o_1 makes a true and nobody other than o_2 makes b true. If a and b are pre-conditions of o_5 and o_8 respectively, then o_5 and o_8 are dynamic mutex at action level $(i + 1)$ if they occur in $(i + 1)$ th action level. Furthermore, if r and s are added only by o_5 and o_8 respectively, then r and s are dynamic mutex in proposition level $(i + 2)$ if o_5 and o_8 are mutex at action level $(i + 1)$.

Graphplan grows a planning graph until it contains a proposition level which contains all goal conditions such that no two of them are mutex. It then begins a backward search on the planning graph to find a plan. If no plan is found, it grows the planning graph more and carries out backward search again. It is clear that if there are a large number of mutually exclusive actions, the size of the planning graph will be higher. This won't be just

because of the higher number of action levels and proposition levels, but also because of lot of actions and propositions in these levels. Since the number of actions at any action level in the planning graph is generally lower than $|O|$, a state-space encoding based on a planning graph is smaller than a naive state-space encoding [Kautz & Selman 1999]. However, it is clear that a state-space encoding based on a planning graph can still be very large when there are a large number of mutually exclusive actions (because of the higher value of k). Note that dynamic mutexes need not be added to a SAT encoding since they follow from static mutexes.

As shown in the lower part of Figure 10, one can remove several consecutive proposition levels and the action levels that precede/succeed them. One can then decide to represent world states at the time points where remaining proposition levels exist. Then, one can introduce y partially ordered steps in the region between two world states such that y is the same as the total number of non-no-op actions contained in the removed action levels which existed between proposition levels which were replaced by world states. Each of the y steps can then be bound to some action from some removed action level. That way, for each action, o_x , from each of the removed action levels, there is a step among y steps which is bound to o_x . Note that each action in Figure 10 deletes one or more of its pre-conditions. Thus, there is no action that can occur at steps i and $(i + 1)$ in a plan. Thus, we have 3 ($\max(2,3)$) and 5 ($\max(4,5)$) steps in the two regions in Figure 10, instead of $(2 + 3)$ and $(4 + 5)$, respectively. Consider the general case of removing n consecutive action levels containing k_1, k_2, \dots, k_n non no-op actions respectively. Let m_1, m_2, \dots, m_n be the respective number of pairs of static mutex actions in these action levels. Since each of the $(k_1 + k_2 + k_3 + \dots + k_n)$

steps in the corresponding region in the unifying encoding has a unique action binding, $(m_1 + m_2 + \dots + m_n)$ clauses that are needed to state static mutexes between actions in the state-space encoding are not needed in the unifying encoding. Since most actions delete one or more of their own pre-conditions, most actions cannot occur at both time steps t and $(t + 1)$ in a plan. Thus, it is enough to have $\frac{(k_1+k_2+k_3+\dots+k_n)}{2}$ steps in a region of the unifying encoding (when n is even). Let this number of steps be s_{1n} , where $1n$ is used in the subscript because the steps are introduced for removed action levels with indices ranging from 1 to n . This means that the number of variables for step-action binding for the unifying encoding is half the number of such variables in the state-space encoding. Let U_{1n} denote the union of pre-conditions of actions from the n removed action levels above. The asymptotic expression for the number of causal links between these s_{1n} steps ($O(s_{1n}^2 \mid U_{1n} \mid)$) is not a realistic predictor of the actual number of causal link variables. This is because the unique action bindings of the s_{1n} steps are known, so the number of causal link variables needed will be much lower. For example, consider two steps p_i and p_j among the region's s_{1n} steps such that their action bindings are o_q and o_r respectively. The causal link variable $p_i \xrightarrow{f} p_j$ should not be included in the encoding if neither the add effect list of o_q nor the pre-condition list of o_r contains f . We have discussed earlier in the section on incremental planning, how such exploitation of the action bindings can reduce encoding sizes. Because of the fewer causal link variables and the exploitation of known action bindings, the number of clauses required for threat resolution will also be low. As a result, the unifying encoding can be smaller than the state-space encoding when they are based on planning graph.



A Unifying Encoding

Figure 10: A hybrid unifying encoding based on a planning graph. For readability, some no-ops are not shown in the action levels of the planning graph. Also, the contents of the 3rd and 4th proposition levels are not shown in the planning graph.

10.2 Centralized Multi-Agent Planning

Consider a planning scenario where the set of ground actions is split into n disjoint subsets O_1, O_2, \dots, O_n which can be respectively executed by n different agents. Let O_g be the subset of highest size among the n subsets. Let the sets of preconditions of actions from O_1, O_2, \dots, O_n be U_1, U_2, \dots, U_n respectively. Let U_g denote the set with the highest size among these. In practice, multiple agents may construct subplans which need to be merged and an agent may be able to execute only a subset of the available actions. Let us assume that these n agents are responsible for generating subplans which need to be merged in a sequential fashion in order to achieve a given goal. By merging in a sequential fashion, we mean all actions in the subplan of the i th agent will precede all actions in the subplan of the $(i + 1)$ th agent. Let us see how the unifying encoding with n regions can be used to solve this planning problem. Steps in the i th region of the encoding can be bound to any of the actions from set O_i . Let us assume that each region contains $\frac{k}{n}$ steps. In this case, the number of clauses needed to state that a step in region i cannot be bound to more than 1 action is $O(\frac{k}{n} | O_i |^2)$. The number of clauses needed to state this for all steps is $O(\frac{k}{n} | O_g |^2)$. This is much less than the $O(k | O |^2)$ clauses needed in the naive unifying, state-space and causal encodings. Also, the number of variables representing step-action bindings ($p_i = o_j$) in the unifying encoding for this kind of planning problem will be $O(\frac{k}{n} | O_g |)$. This is also much less than the $O(k | O |)$ variables needed in the naive unifying, causal, and state-space encodings. Finally, the number of causal link variables and clauses needed to resolve threats in this encoding are $O(\frac{k^2}{n} | U_g |)$ and $O(\frac{k^3}{n^2} | U_g |)$, respectively. These are much less than the $O(\frac{k^2}{p} | U |)$ variables and $O(\frac{k^3}{p^2} | U |)$ clauses in the naive unifying encoding. Since the

entire encoding is solved by one SAT solver, we consider this as centralized planning.

Note that the reduction in size of the unifying encoding that we obtained by exploiting the structure of the centralized multi-agent planning problem can be obtained in state-space encoding as well. We can state that only an action from O_1 can occur at any of the time steps between and including 0 and $\frac{k}{n} - 1$. Similarly, only an action from O_2 can occur at a time step between and including $\frac{k}{n}$ and $\frac{2k}{n} - 1$. Then, the number of variables and clauses in the state-space encoding will be $O(k(|O_g| + |U|))$ and $O(k(|O_g| + |U_g|) + k|O_g|^2)$ respectively. Though this is less than the size of the modified unifying encoding discussed above, the size difference is lower now.

10.3 Ordered Merging of Partial Plans

This can also be referred to as centralized multi-agent plan merging. This differs from the case in Section 10.2 because the plans are partially generated in this case and they need to be generated from scratch in Section 10.2. Let us consider several agents that have partially generated plans that need to be extended and merged with the following restriction - all steps in the plan of agent $i > 1$ succeed all steps in the plan of agent $(i - 1)$ and precede all steps in the plan of agent $(i + 1)$. In such a case, the steps in an individual partial plan and the new steps for extending this plan can be considered as steps in a region of the unifying encoding. Let there be p agents and, thus, p regions in the unifying encoding. Let $\frac{k}{p}$ be the number of steps in the partial plan of each agent. Let each plan be extended by q steps, which is very low in comparison with $\frac{k}{p}$. A state-space encoding for this problem will have $(\frac{k}{p} + 1).q.p$ new steps because of the addition of $(\frac{k}{p} + 1)$ groups, each containing q steps, to the partial plan of each of the p agents. Thus, it will have $O((\frac{k}{p} + 1).q.p.(|O| + |U|))$

variables and $O(\left(\frac{k}{p} + 1\right) \cdot q \cdot p \cdot (|O| + |U| + |O|^2))$ clauses. These are the sizes when the order of the steps in the individual partial plans is not changed. If this order can be changed, the number of clauses and variables will increase since multiple copies (q) of each partial plan will have to be made. Also, as seen in the section on control of non-minimality in plans, highly non-minimal plans may be yielded by solving such encodings with several extra steps.

Let us find the size of the unifying encoding for this problem. The unifying encoding will have only $q \cdot p \cdot |O|$ new step-action binding variables. It will also have $O(q^2 \cdot p \cdot |U|)$ new causal link variables. It will have $O(q \cdot p \cdot |O|^2 + q^3 \cdot p \cdot |U|)$ clauses for enforcing unique action binding for new steps and for resolving threats posed by new steps to causal links between the new steps respectively. Note that the number of these clauses and variables is low when q is low. The number of causal link variables $p_i \xrightarrow{f} p_j$, where only one of p_i or p_j is a new step, will be low since the known action bindings of steps from previously generated partial plans can be used to identify causal link variables that are unnecessary. This shows that the unifying encoding can be smaller than the state-space encoding for this problem.

Note that a causal encoding for this problem will have $O(q^3 \cdot p^3 \cdot |U|)$ clauses for resolving threats to causal links between new steps, posed by new steps. A causal encoding for this problem will have $O(q^2 \cdot p^2 \cdot |U|)$ variables for representing causal links between new steps. It is clear that the causal encoding is larger than the unifying encoding, because of the absence of states.

10.4 Ordered Merging of Plans

This problem is a special case of problem in Section 10.3 where $q = 0$. This case differs from that in Section 10.3 because the plans are fully generated in this case. So agents have synthesized their plans and they need to be only merged, with the same restriction as in the case in Section 10.3. The ordering of steps in individual plans can change while merging, however, steps in the plan of agent $i > 1$ will always succeed steps in the plan of agent $(i - 1)$ and precede steps in the plan of agent $(i + 1)$.

Let us find the size of the unifying encoding for this scenario. Since the action bindings of steps are known, there will be only k step-action bindings in the unifying encoding. Since the steps in individual plans may be reordered while merging, there are $O(\frac{k^2}{p})$ variables for representing partial order on steps. Since the action bindings of steps are known, the number of causal link variables and the number of variables of type $Adds(p_i, f)$, $Needs(p_i, f)$, and $Dels(p_i, f)$ will be very low. Because of this, the number of clauses for resolving threats will also be low. The encoding will have $O(\frac{k^3}{p^2})$ clauses for specifying transitivity of partial ordering on steps in the plans of agents.

A state-space encoding for this problem will have $\frac{k^2}{p}$ variables for representing step-action bindings and $O(\frac{k^2}{p} | U |)$ variables for representing world states. This is because there are $\frac{k}{p}$ copies of $\frac{k}{p}$ step plans and because there are p such plans. These copies are made to allow reordering of steps. It is clear that this can lead to generation of highly non-minimal plans. It is guaranteed that a model of the unifying encoding will not have more than k actions. Such a guarantee cannot be provided for the state-space encoding. The state-space encoding will have $O(\frac{k^2}{p} | U |)$ clauses to encode explanatory frame axioms. Thus, it is clear that the

unifying encoding will have fewer variables and fewer clauses than the state-space encoding in this planning scenario.

10.5 State-based and Causal Domain-Specific Knowledge

It has been shown that domain-specific knowledge can speed up plan synthesis, since the knowledge can be added as constraints to a SAT encoding and propagated to simplify the encodings [Kautz & Selman 1998], [Mali 1999(c), 2000(b)]. The knowledge may be about the states of objects or about the order in which actions can or cannot occur. Roughly, the knowledge can be classified into state-based knowledge and causal knowledge. State-based knowledge is easier to specify in state-space encodings and causal knowledge is easier to specify in causal encodings. By “easier to specify” we mean that fewer variables and fewer clauses are required to specify the knowledge.

Consider the transportation logistics domain for example. In this domain, packages are to be delivered to specified locations using given vehicles. It is clear that a package delivered to its destination location should not be loaded into any vehicle. Consider an encoding with 3 time steps such that D_1, D_2 are respective destinations of packages P_1, P_2 and V_1, V_2 are vehicles in which packages may be loaded. The constraint above can be specified as a conjunction of the following formulae generated for various packages and various time steps - $(at(P_1, D_1, 1) \Rightarrow (\neg in(P_1, V_1, 2) \wedge \neg in(P_1, V_1, 3) \wedge \neg in(P_1, V_2, 2) \wedge \neg in(P_1, V_2, 3)))$. The predicate “ $at(x, y, z)$ ” denotes that object x is at location y at time z . Similarly, “ $in(x, y, z)$ ” denotes that object x is inside object y at time z . For p' packages, k steps in the encoding, and v' vehicles, $O(p'v'k^2)$ such clauses are needed. To specify the same constraint in causal encoding, one will have to state the following for each package P_i and each step p_j - if p_j

adds $at(P_i, D_i)$, where D_i is destination location of P_i , and a step $p_s, s \neq j$ occurs after p_j , then p_s should not have the effect of adding $in(P_i, V_h)$ where V_h is any of the vehicles. This requires $O(p'v'k^2)$ clauses. So the same number of clauses are required to specify this constraint in both causal and state-space encodings. However, the length of each such clause in the state-space encoding is 2 and the length of each such clause in causal encoding is 3. An extra literal is needed in clauses in the causal encoding because of partial order on steps.

If one wants to specify in a state-space encoding that a particular action o_x should occur in a plan only once, then $O(k^2)$ clauses are needed. These have the form $\neg(o_x(t1) \wedge o_x(t2)), t1 \neq t2, t1, t2 \in [0, k - 1]$. This can be specified in a causal encoding with only the k unit clauses that follow - $(p_1 = o_x), \neg(p_j = o_x), j \in [2, k]$.

If one wants to specify that whenever an action o_x occurs in a plan, action o_y should occur immediately after o_x , then $(k - 1)$ clauses are needed in a state-space encoding. They will have the form $(o_x(t) \Rightarrow o_y(t + 1)), t \in [0, k - 2]$. Such constraints can be used in the transportation logistics domain. For example, a truck should move immediately after some package is loaded into it. $O(k^3)$ clauses are needed to specify such constraints in a causal encoding. This is because one has to state that no other step should occur between these steps. It is clear that more clauses are needed for specifying the constraint in the causal encoding than for the state-space encoding.

What if both state-based and step-order related domain-specific constraints are to be added to an encoding? In such a case, the unifying encoding can have lower size than both state-space and causal encodings when state-based knowledge does not have to be specified at each time step. The unifying encoding will have lower size especially when the domain

specific knowledge stated is of local nature, that is, it is only about partially/fully generated individual plans of various agents, that need to be merged in a specific order.

11 Conclusion

Motivated by the challenge [Kambhampati 1997], we synthesized the complete set of hybrid propositional encodings of planning. Arguing that the number of time steps at which the world state is represented is the key factor that can explain the dichotomy in the current space of propositional plan encodings, we showed that varying the number of time steps at which the world state is represented allows us to synthesize a series of encodings. We showed that neither any hybrid instance of the unifying encoding nor any instance of any other hybrid encoding is smallest in domain-independent non-incremental or generalized-incremental planning. Then, we showed that even the smallest encoding can lead to certain types of non-minimality (too many occurrences of actions) in the plans found. Furthermore, we showed that restricting it to prevent this demands more clauses which make it as large as the causal encoding, which is harder to solve in the non-incremental planning scenario and has thus been neglected in the previous research. This points out new tradeoffs in plan encodings.

We also empirically showed that the hybrid encodings are harder to solve in domain-independent non-incremental planning. In a general case, the hybrid encodings neither combine the best of both worlds (lowest size of the state-space encoding and highest flexibility in step reordering in the causal encoding) nor preserve any advantages of either world. Then, we investigated the potential of hybrid encodings for the following specific planning scenarios - large number of mutually exclusive actions, centralized multi-agent planning, ordered

merging of partially generated plans, ordered merging of fully generated plans, and use of both state-based and causal domain-specific knowledge. We showed that hybrid encodings are smallest or are likely to be smallest in these kinds of problems.

Acknowledgement

This work is supported in part by NSF grant IIS-0119630. The author thanks anonymous referees for very useful comments. The author thanks his former advisor Subbarao Kambhampati for useful comments on the previous draft of this paper. The author thanks Mark Iwen for reading the entire paper and making detailed comments. The author thanks David Wilkins, Ichiro Suzuki, Karen Myers, and Peter Haddawy for useful discussions about planning as satisfiability. The author also thanks the following students at the University of Wisconsin, Milwaukee for useful discussions about improving planning as satisfiability - Mark Iwen, Javier Sanchez, Tran Son, William Schroeder, Krishnendu Ghosh, Usha Venkatesh, Michael Reineck, Amulya Antonydoss, Hung Huynh, Jos'e Gutierrez, Jantira Nakhapakorn, Chitra Kashikar and Linlin Zhang.

References

[**Bacchus & Kabanza 2000**] Faheim Bacchus and Froduald Kabanza, Using temporal logics to express search control knowledge for planning, *Artificial Intelligence*, Vol. 16, 2000, pp. 123-191.

[**Baiocchi et al. 1998**] Marco Baiocchi, Stefano Marcugini and Alfredo Milani, C-SATPlan: A SATPlan-based tool for planning with constraints, Working notes of the AIPS (Artificial Intelligence Planning Systems) workshop “Planning as combinatorial search”, Pittsburgh, 1998.

[**Bayardo Jr. & Schrag 1997**] Roberto Bayardo Jr. and R.C. Schrag, Using CSP look-back techniques to solve real world SAT instances, Proceedings of the National Conference on Artificial Intelligence (AAAI), 1997, pp. 203-208.

[**Bedrax-Weiss et al. 1996**] Tania Bedrax-Weiss, Ari Jonsson and Matthew Ginsberg, Unsolved problems in planning as constraint satisfaction, Unpublished manuscript, 1996.

[**van Beek & Chen 1999**] Peter van Beek and Xinguang Chen, CPlan: A constraint programming approach to planning, Proceedings of National Conference on Artificial Intelligence (AAAI), Orlando, 1999, pp. 585-590.

[**Blum & Furst 1995**] Avrim Blum & Merrick Furst, Fast planning via planning graph analysis, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1995, pp. 1636-1642.

[**Bockmayr & Dimopoulos 1999**] Alexander Bockmayr and Yannis Dimopoulos, Integer

programs and valid inequalities for planning problems, Proceedings of European Conference on Planning (ECP), 1999, Editors Susanne Biundo and Maria Fox, Printed as “Lecture notes in Artificial Intelligence 1809” by Springer-Verlag, 2000, pp. 239-251.

[**Brafman 1999**] Ronen I. Brafman, Reachability, Relevance, Resolution and the planning as satisfiability approach, Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), 1999.

[**Brafman & Hoos 1999**] Ronen I. Brafman and Holger H. Hoos, To encode or not to encode - I: linear planning, Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), 1999.

[**Braha & Brafman 1998**] Dan Braha and Ronen Brafman, A SAT encoding for design search, TR-98-02, Dept. of math and computer science, Ben-Gurion university, 1998, 1-13.

[**Crawford & Selman 1996**] J. Crawford and B. Selman, New methods for solving large constraint and reasoning problems, Tutorial presented at National Conference on Artificial Intelligence (AAAI), 1996.

[**Do & Kambhampati 2000**] Minh Binh Do and Subbarao Kambhampati, On the use of LP relaxations in solving the SAT encodings of planning problems, Withdrawn after acceptance at the AAAI workshop on AI and Operations Research, Austin, Texas, 2000.

[Do et al. 2000] Minh Binh Do, Biplav Srivastava and Subbarao Kambhampati, Investigating the effect of relevance and reachability constraints on SAT encodings of planning, Proceedings of International Conference on Artificial Intelligence Planning and Scheduling (AIPS), 2000.

[Ernst et al. 1997] Michael Ernst, Todd Millstein and Daniel Weld, Automatic SAT compilation of planning problems, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997, pp. 1169-1176.

[Erol et al. 1994] K. Erol, J. Hendler and D. Nau, HTN planning: complexity and expressivity, Proceedings of National Conference on Artificial Intelligence (AAAI), 1994

[Ferraris & Giunchiglia 2000] Paolo Ferraris and Enrico Giunchiglia, Planning as satisfiability in nondeterministic domains, Proceedings of National Conference on Artificial Intelligence (AAAI), 2000, pp. 748-753.

[Gerevini & Schubert 1998] Alfonso Gerevini and Lenhart Schubert, Inferring state constraints for domain-independent planning, Proceedings of the National Conference on Artificial Intelligence (AAAI), 1998, pp. 905-912.

[Gerevini & Schubert 2000] Alfonso Gerevini and Lenhart Schubert, Discovering state

constraints in DISCOPLAN: Some new results, Proceedings of National Conference on Artificial Intelligence (AAAI), 2000.

[**Giunchiglia et al. 1998**] Enrico Giunchiglia, Alessandro Massarotto and Roberto Sebastiani, Act and the rest will follow: Exploiting determinism in planning as satisfiability, Proceedings of the National Conference on Artificial Intelligence (AAAI), 1998, pp. 948-953.

[**Giunchiglia 2000**] Enrico Giunchiglia, Planning as satisfiability with expressive action languages: Concurrency, constraints and nondeterminism, Proceedings of Knowledge Representation and Reasoning conference (KR), 2000.

[**Huang et al. 1999**] Yi-Cheng Huang, Bart Selman and Henry Kautz, Control knowledge in planning: Benefits and Tradeoffs, Proceedings of National Conference on Artificial Intelligence (AAAI), 1999, pp. 511-517.

[**Huang et al. 2000**] Yi-Cheng Huang, Bart Selman and Henry Kautz, Learning declarative control rules for constraint-based planning, Proceedings of International Conference on Machine Learning (ICML), 2000.

[**Kambhampati & Hendler 1992**] Subbarao Kambhampati and James Hendler, A validation structure-based theory of plan reuse, Artificial Intelligence 55, 1992.

[**Kambhampati 1995**] Subbarao Kambhampati, Admissible pruning criteria for planning, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[**Kambhampati 1997**] Subbarao Kambhampati, Challenges in bridging the plan synthesis paradigms, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997.

[**Kambhampati 1997(b)**] Subbarao Kambhampati, Refinement planning as a unifying framework for plan synthesis, AI magazine, Summer 1997, pp. 1-56.

[**Kambhampati et al. 1998**] Subbarao Kambhampati, Amol Mali and Biplav Srivastava, Hybrid planning for partially hierarchical domains, Proceedings of National Conference on Artificial Intelligence (AAAI), Madison, pp. 882-888.

[**Kambhampati & Srivastava 1996**] S. Kambhampati and B. Srivastava, Unifying state-space and plan-space approaches to classical planning, Technical Report, Dept. of computer science, Arizona state university, 1996.

[**Kautz 2000**] Henry Kautz, Satisfiability and state-transition systems: An AI perspective, Invited talk at Conference on Automated Deduction (CADE), 2000.

[**Kautz et al. 1996**] Henry Kautz, David McAllester and Bart Selman, Encoding plans

in propositional logic, Proceedings of Knowledge Representation and Reasoning Conference, (KR), 1996, pp. 374-384.

[**Kautz & Selman 1992**] Henry Kautz and Bart Selman, Planning as satisfiability, Proceedings of 10th European Conference on Artificial Intelligence (ECAI), Austria, 1992, pp. 359-363.

[**Kautz & Selman 1996**] Henry Kautz and Bart Selman, Pushing the envelope: Planning, Propositional logic and Stochastic search, Proceedings of the National Conference on Artificial Intelligence (AAAI), 1996, pp. 1194-2001.

[**Kautz & Selman 1997(a)**] Henry Kautz and Bart Selman, Compute-intensive methods in artificial intelligence, Tutorial presented at the National Conference on Artificial Intelligence (AAAI), 1997.

[**Kautz & Selman 1997(b)**] Henry Kautz and Bart Selman, The nature of experimentation in artificial intelligence and computer science, Presented at the workshop on empirical artificial intelligence, held at the International Joint Conference on Artificial Intelligence (IJCAI), 1997.

[**Kautz & Selman 1998(a)**] Henry Kautz and Bart Selman, The role of domain-specific knowledge in planning as satisfiability framework, Proceedings of the AIPS conference (ar-

tificial intelligence planning systems), 1998, pp. 181-189.

[**Kautz & Selman 1998(b)**] Henry Kautz and Bart Selman, BLACKBOX: A new approach to the application of theorem proving to problem solving, Working notes of the AIPS (artificial intelligence planning systems) workshop “Planning as combinatorial search”, Pittsburgh, 1998.

[**Kautz & Selman 1999**] Henry Kautz and Bart Selman, Unifying SAT-based and graph-based planning, Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), 1999 .

[**Kautz & Walser 1999**] Henry Kautz and Joachim Walser, State-space planning by integer optimization, Proceedings of National Conference on Artificial Intelligence (AAAI), 1999, pp. 526-533.

[**Li 2000**] Chu Mil Li, Integrating equivalency reasoning into Davis-Putnam procedure, Proceedings of National Conference on Artificial Intelligence (AAAI), 2000, pp. 291-296.

[**Li & Anbulagan 1997**] Chu Min Li and Anbulagan, Heuristics based on unit propagation for satisfiability problems, Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), 1997, Vol. 1.

[**Majercik & Littman 1998**] Stephen Majercik and Michael Littman, MAXPLAN: A new approach to probabilistic planning, Proceedings of the international conference on artificial intelligence planning systems (AIPS), 1998, pp. 86-93.

[**Mali 1998**] Amol D. Mali, Refinement-based planning as satisfiability, Proceedings of National Conference on Artificial Intelligence (AAAI), Student abstract, 1998, pp. 1194.

[**Mali & Kambhampati 1998(a)**] Amol Mali and Subbarao Kambhampati, Encoding HTN planning in propositional logic, Proceedings of international conference on Artificial Intelligence Planning Systems (AIPS), 1998, pp. 190-198.

[**Mali & Kambhampati 1998(b)**] Amol Mali and Subbarao Kambhampati, Frugal propositional encodings for planning, Notes of Workshop “Planning as combinatorial search”, Pittsburgh, June 1998, pp. 89-94.

[**Mali & Kambhampati 1998(c)**] Amol Mali and Subbarao Kambhampati, Refinement-based planning as satisfiability, Notes of Workshop “Planning as combinatorial search”, Pittsburgh, June 1998, pp. 95-100.

[**Mali 1999(a)**] Amol Mali, Hybrid propositional encodings of planning, Proceedings of National Conference on Artificial Intelligence (AAAI), 1999, Student abstract, pp. 972.

[**Mali 1999(b)**] Amol Mali, Plan merging and plan reuse as satisfiability, Susanne Biundo and Maria Fox (eds), Recent advances in AI planning, Lecture notes in artificial intelligence 1809, Springer-Verlag 2000, 5th European conference on planning (ECP), Durham, UK, Sept. 1999, pp. 84-96.

[**Mali 1999(c)**] Amol Mali, Hierarchical task network planning as satisfiability, Susanne Biundo and Maria Fox (eds), Recent advances in AI planning, Lecture notes in artificial intelligence 1809, Springer-Verlag 2000, 5th European conference on planning (ECP), Durham, UK, Sept. 1999, pp. 122-134.

[**Mali & Kambhampati 1999**] Amol Mali and Subbarao Kambhampati, On the utility of causal encodings, Proceedings of National Conference on Artificial Intelligence (AAAI), Orlando, 1999, pp. 557-562.

[**Mali 2000(a)**] Amol Mali, On the hybrid propositional encodings of planning, Editor M. H. Hamza, Proceedings of the International conference on artificial intelligence and soft computing (ASC), July 24-26 2000, Banff, Alberta, Canada, pp. 334-342.

[**Mali 2000(b)**] Amol Mali, Enhancing HTN planning as satisfiability, Editor M. H. Hamza, Proceedings of the International conference on artificial intelligence and soft computing (ASC), July 24-26 2000, Banff, Alberta, Canada, pp. 325-333.

[**Mali 2001**] Amol Mali, Recent advances in efficient plan synthesis, Half-day tutorial presented at the IEEE International Conference on Robotics and Automation (ICRA), Seoul, Korea, 21 May 2001.

[**McCain & Turner 1998**] Norman McCain and Hudson Turner, Satisfiability planning with causal theories, Proceedings of the Knowledge Representation & Reasoning conference (KR), 1998.

[**Milani et al. 1998**] Alfredo Milani, Stefano Marcugini and Marco Bairoletti, Partial plans completion with Graphplan, Working notes of the workshop “Planning as combinatorial search”, held at the International Conference on the Artificial Intelligence Planning Systems (AIPS), 1998.

[**Rintanen 1998**] Jussi Rintanen, A planning algorithm not based on directional search, Proceedings of the Knowledge Representation & Reasoning conference (KR), 1998.

[**Smith & Peot 1996**] David Smith and Mark Peot, Suspending recursion in causal link planning, Proceedings of the international conference on Artificial Intelligence Planning Systems (AIPS), 1996.

[**Vossen et al. 2000**] Thomas Vossen, Michael Ball, Amnon Lotem and Dana Nau, Applying integer programming to AI planning, Knowledge Engineering Review, 2000.

[**Wolfman & Weld 2000**] Steven A. Wolfman and Daniel S. Weld, Combining linear programming and satisfiability solving for resource planning, Special issue of Knowledge Engineering Review, 2000.

List of Figures

Figure 1: Interactions between steps in propositional encodings of classical planning problems.

Figure 2: The schemas for generating the unifying encoding.

Figure 3: In the absence of the state representation more causal link variables are required and, thus, more possibilities of threats exist.

Figure 4: Incremental planning with a contiguous plan by making multiple copies of the old plan and including multiple blocks of steps for new actions ($k = 2$, $q = 3$).

Figure 5: A comparison of the sizes of encodings for incremental planning. A plan of k steps is included in the encodings to solve a new problem in an incremental style. Upto q new actions may be added and old actions may be removed.

Figure 6: Completion of a contiguous plan.

Figure 7: A comparison of the encodings for the plan completion problem.

Figure 8: A comparison of the encodings for the incremental planning scenario with a re-

restriction on the number of occurrences of an action that occurs s times in an old plan. The size of the state-space encoding in this table is based on the assumption that the encoding is generated by scheme 2 in Section 7.1.2. If the state-space encoding based on scheme 1 in Section 7.1.1 is used, $O((k^2 + (k + 1).q). | U | + (k + 1).q. | O |)$ variables and $O((k^2 + (k + 1).q). | U | + k^3 + k^2.q + (k + 1).q. | O |^2)$ clauses are needed. As shown in section 9, another way to state the restriction about the number of occurrences of an action reduces the number of clauses in the unifying encoding, with $p = \frac{k}{q}$, from $O((k + p.q)^{(q+s+1)})$ to $O(\frac{k^2}{q})$.

Figure 9: Empirical results on various (hybrid) instances of the unifying encoding from Figure 1. $p = 1$ corresponds to the purely causal encoding and $p = k$ corresponds to the purely state-space encoding. V, C, T denote the number of variables, clauses and time needed to solve the encodings respectively. Times are in CPU seconds. A “*” indicates that the encoding was not solved within 5 minutes of CPU time. A “-” denotes that the encoding was not generated since the number of regions was not an integer. The descriptions of the benchmark domains used and the “satz” systematic SAT solver used are available at <http://www.cs.yale.edu/HTML/YALE/CS/HyPlans/~mcdermott.html> and <http://aida.intellektik.informatik.th-darmstadt.de/~hoos/SATLIB> respectively. Log is the transportation logistics domain and Tsp is the traveling salesperson domain.

Figure 10: A hybrid unifying encoding based on a planning graph. For readability, some no-ops are not shown in the action levels of the planning graph. Also, the contents of the 3rd and 4th proposition levels are not shown in the planning graph.