

Hierarchical Task Network Planning As Satisfiability*

Amol Dattatraya Mali

Dept. of Elect. Engg. & Computer Science, P.O.Box 784,
University of Wisconsin, Milwaukee, WI 53201, USA
mali@miller.cs.uwm.edu

July 2, 2003

Abstract

The satisfiability paradigm has been hitherto applied to planning with only primitive actions. On the other hand, hierarchical task networks have been successfully used in many real world planning applications. Adapting the satisfiability paradigm to hierarchical task network planning, we show how the guidance from the task networks can be used to significantly reduce the sizes of the propositional encodings. We report promising empirical results on various encodings that demonstrate an orders of magnitude reduction in the solving times.

1 Introduction

Given a planning problem $\langle I, G, O \rangle$, where I is the initial state of the world, G is the goal state and O is the set of executable actions, [Kautz & Selman 96]

*This paper appears in the proceedings of European Conference on Planning (ECP), Durham, UK, Sept. 1999, pp. 122-134. I thank Subbarao Kambhampati and the anonymous referees of ECP-1999 for useful comments on this work. The college of engineering and applied sciences at Univ. of Wisconsin, Milwaukee provided financial support for attending the conference and presenting the work.

showed that finding a k -step solution to this problem can be cast as propositional satisfiability. The basic idea is to generate a propositional formula (called an *encoding*) such that any model of this formula will correspond to a k step solution to the original problem. The clauses in the encoding thus capture various constraints required for proving that a k -length action sequence is a solution to the planning problem. Promising results were obtained by using this paradigm, as shown by [Kautz & Selman 96],[Ernst et al 97] and [Giunchiglia et al 98].

However this paradigm has been applied to planning with only primitive actions. On the other hand, HTN planners have been used in several fielded applications including space platform construction, satellite planning and control [Tate 77], beer factory production line scheduling, military operations planning [Wilkins 88], image processing for science data analysis and deep space network antenna operations [Estlin *et al.* 97]. Casting HTN planning as satisfiability is also one of the challenges proposed recently [Kambhampati 97]. Though the satisfiability paradigm has been adapted to handle HTN planning [Mali & Kambhampati 98], no empirical results on the performance of the encodings of HTN planning are available. [Mali & Kambhampati 98] describe three encodings of HTN planning and report the number of clauses and variables that these encodings contain. They show that the HTN encodings contain more clauses and more variables than the action-based encodings of [Kautz et al 96]. It has thus remained unclear whether the HTN encodings are easier to solve. This is the question that has motivated our work. Our work makes the following contributions.

- We provide a procedure for pre-processing the causal HTN encodings of [Mali & Kambhampati 98] and show that it reduces the sizes of the encodings significantly and that these smaller encodings are also faster to solve.
- We modify the HTN encodings of [Mali & Kambhampati 98], so that their models obey the *criterion of parsing* (the models of the HTN encodings must be same as the the plans that can be parsed by the task reduction schemas).
- We provide an empirical evaluation of the HTN encodings of [Mali & Kambhampati 98] to illustrate the effectiveness of our pre-processing procedure and show that the pre-processed causal HTN encodings can

be smaller than the smallest of the action-based encodings and also the fastest to solve, on some problems.

This paper is organized as follows. In section 2, we discuss the basics of the HTN encodings and explain some notation used to represent the constraints from the task reduction schemas, in the encodings. In section 3, we discuss procedures of pre-processing the HTN encodings and discuss how they reduce the sizes of the encodings. In section 4, we show how the HTN encodings can be augmented with constraints to obey the criterion of parsing, without significantly increasing their size. We also show how these constraints avoid the generation of non-minimal plans. In section 5, we describe an empirical evaluation of the HTN encodings and the action-based encodings and discuss the insights obtained. We report conclusions in section 6.

2 Basics of HTN Encodings

The propositional encodings for HTN planning can be developed by constraining the action-based encodings of [Kautz et al 96], such that their satisfying models also conform to the grammar of solutions of interest to a user, specified by the task reduction schemas. Since most implemented HTN planners [Wilkins 88][Tate 77] share a lot of structure of the partial order planners, the “causal encodings” in [Kautz *et al.* 96] provide a natural starting place for developing encodings for HTN planning. Since HTN planning can be viewed as just an augmentation of action-based planning with a grammar of legal solutions, there is no reason to constrain ourselves to causal encodings. Indeed, an HTN encoding based on the state-based encoding of [Kautz *et al.* 96] also exists, as shown by [Mali & Kambhampati 98]. The representation we use for the task reduction schemas is consistent with that used in [Erol 95].

In what follows, we assume that o_i denotes a STRIPS style ground primitive action. p_i denotes a primitive step that is mapped to a ground primitive action or the null action (no-op) ϕ that has no pre-conditions and no effects. O denotes the set of all ground primitive actions from the domain.

N_i denotes a ground non-primitive task. d_i denotes the number of reduction schemas of N_i . s_i denotes a non-primitive step that is mapped to one and only one non-primitive task. r_{ij} denotes j th reduction schema of N_i ,

since a task may be reduced in multiple ways. Each reduction schema r_{ij} may contain the following types of constraints (that capture the causal links and orderings between non-primitive tasks and/or primitive actions) - (i) mapping from an action symbol to a primitive action, for example, $o_1 : load(x, l_1, R)$ (ii) mapping from a non-primitive task symbol to non-primitive task name, for example, $N_2 : achieve(at(R, l_2))$ (iii) $o_s \prec N_p$ (where \prec denotes temporal precedence) (iv) $N_p \prec o_s$ (v) $o_p \xrightarrow{f} o_q$, which denotes the causal link where o_p is the contributor and o_q is the consumer. (vi) $? \xrightarrow{f} o_p$ (which denotes that the contributor is not known a priori, and will be introduced in the partial plan as planning progresses, possibly by the reduction of some other task) (vii) $o_q \xrightarrow{f} ?$ (viii) $N_q \xrightarrow{f} o_p$ (ix) $o_p \xrightarrow{f} N_q$ (x) $N_p \prec N_q$ (xi) $N_p \xrightarrow{f} N_q$ (xii) $o_p \prec o_q$ (xiii) $? \xrightarrow{f} N_q$ and (xiv) $N_p \xrightarrow{f} ?$. The semantics of these constraints is discussed in [Mali & Kambhampati 98].

Mapping a step s_k to N_i (denoted by $s_k = N_i$) means choosing N_i to solve the planning problem. In this case, it is necessary to carry out N_i by satisfying the constraints in some reduction schema of N_i . $t(r_{ij})_k$ then represents a conjunction of the disjunction of all potential ways of satisfying each constraint in r_{ij} . m denotes the number of non-primitive tasks used in an HTN encoding. M_i denotes the maximum number of primitive actions in reduction schema of N_i and the reduction schema containing M_i primitive actions is denoted by r_{iJ} . $M = \max(\{M_i \mid i \in [1, m]\})$.

When s_k is mapped to N_i and the reduction schema r_{ij} is used to reduce N_i , primitive steps ranging from $p_{a'_{ijk}}$ to $p_{a''_{ijk}}$ are mapped to the primitive actions from r_{ij} , where,

$$a'_{ijk} = (k - 1) * M + 1, a''_{ijk} = a'_{ijk} + b_{ij} - 1$$

where b_{ij} is the number of primitive actions in r_{ij} .

The total number of primitive steps in an HTN encoding are T , where $T = M * K$, K being the number of non-primitive steps chosen. This can be viewed as allocating M primitive steps ranging from p_{i*M} to $p_{(i+1)*M-1}$ to each non-primitive step s_i (M is computed automatically by examining the reduction schemas and is not supplied by a user). Such an allocation done a priori does not affect the soundness and completeness of an HTN encoding, since all potential orderings between the primitive steps are represented and the primitive steps can be interleaved as per need. A plan may contain less

than T primitive actions and less than K reduction schemas may be required to synthesize the plan. Such plans can be found by mapping the excess steps to ϕ .

We refer to the encodings that are not pre-processed or simplified as “naive” encodings. All potential ways of satisfying the constraints in the reduction schemas and all potential choices of the reduction schemas are represented in the encodings, to achieve soundness and completeness. Note that we do not handle recursive task reduction schemas.

2.1 Causal HTN Encodings

The task reduction schemas can be used to control planning either in a top-down or a bottom-up way. In the top-down way, which is followed in most implemented HTN planners, planning starts with non-primitive tasks, and the reduction schemas are used to gradually reduce them into more concrete actions (while taking care of ensuing interactions), as in [Erol 95]. In the bottom-up way [Barrett & Weld 94], the (partial) solutions generated by an action-based planner are incrementally parsed with the help of reduction schemas and the branches leading to solutions that cannot be parsed are pruned. Two causal HTN encodings that are motivated by these notions of decomposition and parsing check are proposed in [Mali & Kambhampati 98]. We review them next. Note that our top-down and bottom-up HTN encodings differ only in the presence of certain variables and clauses. The encodings are not directional and they are not necessarily solved in the top-down or bottom-up styles.

Top-Down Causal HTN Encoding - The notion of task decomposition is captured in this encoding with the constraint $\bigwedge_{i=1}^m \bigwedge_{k=1}^K ((s_k = N_i) \Rightarrow (\bigvee_{j=1}^{d_i} t(r_{ij})_k))$. Consider the constraint $N_p \prec N_q$ from the reduction schema of a non-primitive task to which the step s_k is mapped. To represent this, one of the constraints we specify is,

$$\bigvee_{w_1=1, w_1 \neq k}^K \bigvee_{w_2=1, w_2 \neq w_1, w_2 \neq k}^K ((s_{w_1} = N_p) \wedge (s_{w_2} = N_q) \wedge (\bigwedge_{u_1=a'_{pJw_1}}^{a''_{pJw_1}} \bigwedge_{u_2=a'_{qJw_2}}^{a''_{qJw_2}} (p_{u_1} \prec p_{u_2})))$$

(since there must exist non-primitive steps that are mapped to N_p and N_q respectively and each primitive step mapped to an action in the reduction of N_p must precede each primitive step mapped to an action in the reduction

of N_q). By introducing $(K - 1) * (K - 2)$ new “intermediate” variables, each implying a conjunction in the disjunction above, we can reduce the number of clauses there from exponential to $O(K^2 * M_p * M_q)$.

Bottom-Up Causal HTN Encoding - The non-primitive step and non-primitive task symbols are not used in the bottom-up encoding. It uses only primitive steps that are mapped to actions. The notion of the allocation of certain number of primitive steps to the reduction of a task is however present in the bottom up encoding as well. Thus the number of primitive steps in the bottom-up encoding is $T(K * M)$. The bottom up encoding contains the constraint that some $t(r_{ij})_k$, $k \in [1, K], i \in [1, m], j \in [1, d_i]$ must be true, to ensure that the constraints from some reduction schemas are respected by the solution.

Due to the absence of s_i, N_j symbols, different number of clauses and variables are required to represent certain constraints in the bottom-up HTN encodings than the top-down HTN encodings. For example, consider the constraint $N_p \prec N_q$ (contained in r_{ij}) that needs to be true when $t(r_{ij})_k$ is true. To represent this in the encoding, one of the constraints we specify is,

$$\bigvee_{z_1=1}^{d_p} \bigvee_{z_2=1}^{d_q} \bigvee_{w_1=1, w_1 \neq k}^K \bigvee_{w_2=1, w_2 \neq k, w_1 \neq w_2}^K (t(r_{pu_1})_{w_1} \wedge t(r_{qu_2})_{w_2} \wedge (\bigwedge_{u_1=a''_{pz_1 w_1}}^{a''_{pz_1 w_1}} (\bigwedge_{u_2=a'_{qz_2 w_2}}^{a'_{qz_2 w_2}} (p_{u_1} \prec p_{u_2}))))))$$

This requires $O(K^2 * M_p * M_q * d_p * d_q)$ clauses and $O(K^2 * d_p * d_q)$ intermediate variables, both being higher than those required in the top-down causal encoding by the factor of $O(d_p * d_q)$. This difference in the number of clauses and intermediate variables also applies to the representation of the constraint $N_p \xrightarrow{f} N_q$.

The causal HTN encodings contain $O(K * m * d * Z + T^3 * |\Gamma|)$ clauses and $O(K * m * d * V + T^2 * |\Gamma|)$ variables, Z, V being the number of clauses and variables used to represent all potential ways of satisfying all the constraints in a reduction schema of a non-primitive task. $d = \max(\{d_i \mid i \in [1, m]\})$. Γ denotes the set of ground pre-conditions of actions in the domain. An action-based causal encoding with T primitive steps, without the constraints the reduction schemas, contains $O(T^3 * |\Gamma|)$ clauses (to resolve all potential threats to all potential causal links) and $O(T^2 * |\Gamma|)$ variables (all potential causal links), showing that the causal HTN encodings contain more clauses and more variables than the action-based causal encoding of [Kautz et al 96].

2.2 State-based HTN Encoding

To synthesize this encoding, [Mali & Kambhampati 98] constrain the state-based encoding of [Kautz et al 96] with the constraints from the task reduction schemas. This encoding contains $T(K * M)$ contiguous steps. In this encoding, $o_i(t)$ denotes that the primitive action o_i occurs at time t . $f_i(t)$ denotes that the proposition f_i is true at time t .

Consider the causal link $o_p \xrightarrow{f} o_q$. We convert this causal link into an ordering and an interval preservation constraint, as shown below.

$$\bigvee_{i=0}^{T-2} \bigvee_{j=i+1}^{T-1} (o_p(i) \wedge o_q(j) \wedge (\bigwedge_{t=i+1}^j f(t)))$$

U denotes the set of ground pre-condition and effect propositions in the domain. The state-based HTN encoding contains $O(b' * d^2 * m * T^3 * |O| + T * (|O| + |U|))$ clauses and $O(b' * d^2 * m * T^2 * |O| + T * (|O| + |U|))$ variables respectively, where $b' = \max(\{b_{ij} \mid i \in [1, m], j \in [1, d_i]\})$, higher than the number of clauses ($O(T * (|O| + |U|))$) and variables ($O(T * (|O| + |U|))$) in the action-based state-based encoding of [Kautz et al 96] (these expressions of [Kautz et al 96] follow from the fact that the world state is represented at each time step (in the state-based encodings) and an action occurring at time t implies the truth of its pre-conditions at t and the truth of its effects at $(t + 1)$. Explanatory frame axioms are required to insist that if $(f(t) \wedge \neg f(t + 1))$ is true, some action deleting f must occur at t).

3 Pre-processing

In this section, we show how the HTN encodings can be pre-processed to reduce their sizes.

3.1 Causal HTN Encodings

Consider the allocation of primitive steps to the non-primitive steps in Fig. 1. Each non-primitive step may be mapped to either N_1 or N_2 which have 3 and 2 reductions respectively. The primitive actions from the reduction schemas of these tasks are also shown there. These actions may have ordering constraints between them and there may be additional constraints in the reduction schemas (not shown in Fig. 1). Since the primitive steps in the

causal encodings are partially ordered and all such orderings are represented, one can even fix the potential step→action mapping ($p_i = o_j$) a priori, in the form of a disjunction of step→action mappings e.g. One can infer that $((p_1 = o_1) \vee (p_1 = o_3) \vee (p_1 = o_6) \vee (p_1 = \phi) \vee (p_1 = o_9) \vee (p_1 = o_{13}))$. In particular, an i th primitive step allocated to a non-primitive step s_j will be mapped to either ϕ or only the i th primitive action in some reduction schema of some non-primitive task, rather than any of the $|O|$ actions in the domain (which is the case for the naive encodings). For each $p_i, i \in [1, T]$, a naive encoding represents the mapping $((\bigvee_{j=1}^{13} (p_i = o_j)) \vee (p_i = \phi))$. We refer to the reduction in the domain of p_i as the reduction in the disjunction in the step→action mapping. This reduction can be propagated to significantly reduce the sizes of the causal HTN encodings. Hence the sizes reported by [Mali & Kambhampati 98] should not be interpreted to conclude that in practice, the HTN encodings will always be larger than the corresponding action-based encodings.

The naive causal encodings contain all $O(T^2 * | \Gamma |)$ potential causal links. However, with the reduced disjunction in the step→action mapping, before generating any causal link $p_i \xrightarrow{f} p_j$, one can check (a) if p_i may be mapped to an action that adds f and (b) if p_j may be mapped to an action that needs f . If either (a) or (b) is false, this causal link variable need not be created. If this link is not created, no clauses need to be generated to resolve threats to this link. Even if (a) and (b) are both true, before generating the threat resolution clause $((p_i \xrightarrow{f} p_j) \wedge Del_s(p_s, f)) \Rightarrow ((p_s \prec p_i) \vee (p_j \prec p_s))$, one can check if p_s may be mapped to an action that deletes f . If this is not the case, this clause need not be generated. Action-based causal encodings contain $O(T * |O|^2)$ clauses to specify that a step cannot be bound to more than one action. With the reduced disjunction in the step→action mapping, many of these $O(T * |O|^2)$ clauses will not have to be generated. Though we have discussed here how only the dominant terms in the sizes of the causal HTN encodings are lowered by the propagation of the reduced disjunction in the step→action mapping, other less dominant terms are lowered as well. Though the bottom-up causal HTN encoding does not contain the s_i and N_i symbols, it does contain the $t(r_{ij})_k$ symbols and such a reduction in the disjunction in the step→action mapping is doable there as well. We have integrated this pre-processing with the code that generates the causal HTN encodings. Most current SAT-based planners generate encodings and then

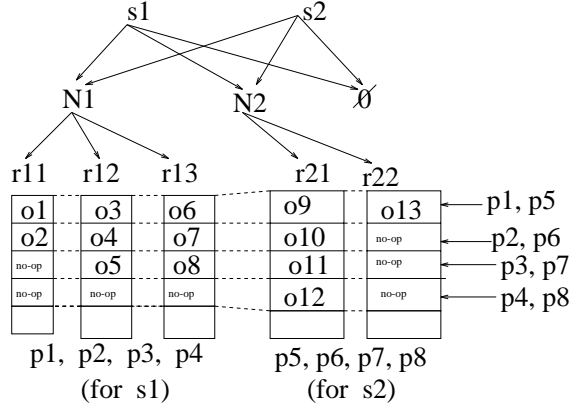


Figure 1: The step allocation in the top-down causal HTN encoding, $K = 2$, $m = 2$, $M = 4$, $T = K * M = 8$

simplify them. However many encodings are too large to store (as shown in Fig. 2), making it impossible to simplify them later. We generate a simplified encoding rather than simplifying a generated encoding, using our pre-processing procedures.

Is the reduction in the disjunction in the step→action mapping always guaranteed? A closer examination shows that when there are at least θ reduction schemas, such that $\theta \geq |O|$, such that i th reduction schema ($1 \leq i \leq |O|$) contains γ occurrences of o_i , $\gamma = M$, any primitive step in the encoding may be mapped to any of the $|O|$ actions in the domain, like the naive action-based encodings, without yielding a reduction in the disjunction in the step→action mapping. In all other cases, there will be a reduction in the disjunction in the step→action mapping.

3.2 State-based HTN Encoding

We use the planning graph [Blum & Furst 95] to frugally represent the potential ways of satisfying the constraints from the reduction schemas. The planning graph is grown starting from the initial state (0th proposition level), by applying all actions whose pre-conditions are satisfied in I , computing the next proposition level (which is a union of the propositions in the previous proposition level and the add and delete effects of the actions applicable in the previous proposition level) and repeating this process. The constraint

$o_2 \prec o_4$ is represented in the naive encoding as $\bigvee_{i=0}^{T-2} (o_2(i) \wedge (\bigvee_{j=i+1}^{T-1} o_4(j)))$. However by examining the actions in the planning graph, one can know the time steps at which o_2 and o_4 may occur and use this information to represent $o_2 \prec o_4$ with fewer variables and fewer clauses. Similarly, the causal links can be represented with fewer variables and fewer clauses.

4 Properties of the Models

In this section, we discuss how the HTN encodings of [Mali & Kambhampati 98] can be constrained, so that their models obey the parsing criterion. In the causal HTN encodings, in case the total number of reduction schemas chosen to solve the problem is less than K (and since the number of primitive steps allocated is intended for the use of K reduction schemas), the extra primitive steps may be mapped to non-null actions, yielding non-minimal plans (plans from which some actions can be removed, preserving the plan correctness). To prevent this, we include clauses in these encodings which state that such excess primitive steps must be mapped to ϕ . That is, (1) if all the $t(r_{ij})_k$ variables, $i \in [1, m], j \in [1, d_i]$ are false, the corresponding M primitive steps allocated to s_k must be mapped to ϕ . (2) Also, if $t(r_{ij})_k$ is true, the excess $(M - b_{ij})$ primitive steps must also be mapped to ϕ . Specifying these restrictions requires $O(K * M * m * d)$ clauses. These prevent non-minimal plans. By mapping the redundant steps to ϕ , we also prevent them from being mapped to actions outside the reduction schemas used, enforcing the criterion of parsing.

Though the steps that will be mapped to the primitive actions from a reduction schema are not known a priori, an allocation of M steps to a reduction schema exists in the state-based HTN encoding as well. Thus the state-based HTN encodings too can be augmented with extra constraints to prevent non-minimality. However, since the explanatory frame axioms allow parallel actions, the models of the state-based encodings can still be non-minimal plans and violate the parsing criterion, unless they are constrained with the only one action at a step restriction.

5 Discussion

Reviewing the HTN encodings of [Mali & Kambhampati 98], we proposed pre-processing methods to reduce their sizes. To test the effectiveness of our methods, we conducted an empirical evaluation of the encodings on several benchmark domains.¹ Only those task reduction schemas that were required for solving the problems were used in generating the encodings. The encodings were generated and solved on a Sun Ultra with 128 M RAM.

The descriptions of the house building and tire world domains were respectively obtained from <http://www.aiai.ed.ac.uk/~oplan/> and <http://pecan.srv.cs.cmu.edu/afs/cs.cmu.edu/local/mosaic/common/omega/Web/People/avrim/graphplan.html>. The encodings were solved with the “satz” solver, except those in the Tsp (traveling sales person domain) which were solved with version 35 of Walksat.² All the encodings were simplified by unit propagation and processed using procedures from section 3, before being solved.³ Like [Ernst et al 97], we do not report the times required to simplify the encodings, since these times were small. A comparison of the sizes of the causal HTN encodings in Fig. 2 (pre-processed and naive) shows that our pre-processing procedure (which is based on the propagation of the reduction in the disjunction in the step→action mapping) reduces both the number of clauses and the variables by orders of magnitude. This also yields an orders of magnitude improvement in the times required to solve the encodings. The sizes of the top-down and bottom-up causal HTN encodings (whether they are pre-processed or naive does not matter) and even their solving times are very close, since the bottom-up causal HTN encodings differed from the top-down causal HTN encodings only in the presence of non-primitive step and non-primitive task symbols.

The HTN encodings contained T ($K * M$) primitive steps and this was generally larger than the number of actions in the plans, as can be seen from Fig. 2. The action-based encodings however contained only k primitive steps, k being the number of actions in the plans, $k \ll T$ (thus the HTN encodings had $(T - k)$ redundant primitive steps, this also shows that the HTN encod-

¹Available at <http://www.cs.yale.edu/HTML/YALE/CS/HyPlans/mcdermott.html>

²The solvers are available at <http://aida.intellektik.informatik.th-darmstadt.de/~hoos/SATLIB>.

³The unit propagation code used is from the Feb. 96 version of the Satplan system (/satplan/system/simplify/unitprop) available at <ftp://ftp.research.att.com/dist/ai>.

ings will have no redundant primitive steps, if the domain knowledge is partitioned among the reduction schemas such that $M = \frac{k}{K}$). The action-based causal encodings were significantly larger than the pre-processed causal HTN encodings, despite the fewer primitive steps used, illustrating the power of the pre-processing method from section 3.1. Also, we used only those actions that appeared in the reduction schemas, in the action-based encodings. Since this will generally not be the case in practice (the number of primitive actions in the reduction schemas will be far lower than the number of primitive actions in the domain), the action-based encodings will be far larger.

We found that the sizes of the state-based HTN encodings pre-processed using the planning graph information and the naive state-based HTN encodings, as well as their solving times did not differ significantly, on most of the problems. This is because any action that occurs at action level i in the plan graph also occurs at all j th action levels, $j > i$ and the sizes of the encodings do not reduce significantly, due to this redundancy. On most of the problems, the state-based HTN encodings still had fewer clauses and fewer variables than both the naive and the pre-processed causal HTN encodings. However the number of variables in the state-based HTN encodings was higher than those in both the naive and pre-processed causal HTN encodings, on problems from the house building and tire world domains. This is because the reduction schemas in these domains contained lot of causal links and precedence constraints and the representation of these constraints on the contiguous steps required more clauses and more variables. Because of this, the state-based HTN encodings were larger and also harder to solve, than the state-based action-based encodings and the causal pre-processed HTN encodings, on these problems.

The state-based action-based encodings were the smallest on most of the problems, however the plans found by solving them, as well as those found by solving the state-based HTN encodings were highly non-minimal, both because the number of steps in the encodings (action-based) were higher (in some cases) than the length of the shortest plan and because the explanatory frame axioms allow parallel actions. For example, the plan found using the state-based action-based encoding, in the tire world had 12 redundant actions and the plan found using the state-based HTN encoding (for the same problem) had 9 redundant actions, because of the cyclicity possible, e.g. *open(container)*, *close(container)*, *do_up(nut, hub)*, *undo(nut, hub)*, *jack_up(hub)*, *jack_down(hub)*, *tighten(nut, hub)* and *loosen(nut, hub)* etc.

The models of the state-based HTN encoding and the state-based action-based encodings for the logistics problem (for which a 21 action plan exists), contained 49 and 42 redundant flights respectively, e.g. $fly(R_1, London, Paris)$ and $fly(R_1, Paris, London)$ etc. Similarly, the model of the state-based HTN encoding for the problem from the Ferry domain (for which a 19 action plan exists) contained 8 redundant actions, e.g. $sail(a, b)$, $sail(b, a)$. The causal encodings did not suffer from this problem of non-minimality of the plans, showing that their models conform to the user intent better than the models of the state-based encodings. To guarantee minimal plans using the state-based HTN encodings, the restriction that only one action should occur at a time step needs to be added to them. The number of non-primitive steps K we used in the encodings was same as the number of reduction schemas required to solve the problems. If the value of K chosen is larger than the necessary, or the reduction schemas supplied themselves contain redundancy, the models of the HTN encodings may still be non-minimal plans.

The causal encodings have been shown to be hard to solve in the action-based planning scenario [Mali & Kambhampati 99]. Our pre-processing method was so effective, that in some domains like the tire world, the pre-processed causal HTN encodings were the smallest (smaller than even the state-based encodings with explanatory frame axioms which have been found to be the smallest in the action-based planning scenario [Ernst et al 97] and [Giunchiglia et al 98]). The structure in Fig. 1 that shows the potential step \rightarrow action mappings can be modified using some soundness and completeness preserving heuristics (without changing the task reduction schemas and without violating the user intent), to yield a larger reduction in the encoding size and we intend to perform an empirical study of the performance of such heuristics in future.

We want to stress that though we compared the performance and sizes of the action-based encodings and the HTN encodings to see if the information from the HTNs could be used to make the encodings easier to solve, efficiency is not always the primary reason for using the HTNs. Since the HTNs provide a grammar of the solutions desired by the users, it is some times necessary to use them to respect the user intent, even if plan synthesis is not faster.

[Kautz & Selman 98] have shown that adding domain specific knowledge to the action-based encodings (though it increases the size of an encoding) can be used to solve them faster. This is because the domain specific knowledge can be efficiently propagated to reduce the size of an encoding. Our work

Domain, # Plan Actions	TDCP	TDCN	BUCP	BUCN	AC	SBH	AS
	V, C, T	V, C, T	V, C, T	V, C, T	V, C, T	V, C, T	V, C, T
Ferry, 19	1715	19753	1699	19737	8535	1242	588
K = 2, T = 30	31162	526697	30687	526657	138172	4856	2436
k = 19	11.31	*	11.42	*	*	60.81	3.96
Ferry, 39	6141	131893	6115	131867	57880	4577	2178
K = 2, T = 60	249153	7438877	247318	7438812	2067472	21406	11431
k = 39	*	-	*	-	*	*	*
Ferry, 27	3153	48985	3133	48965	21361	2324	1104
K = 2, T = 42	85220	1886741	84309	1886691	511168	9820	5080
k = 27	45.96	-	45.63	-	*	*	*
Ferry, 31	4051	70765	4029	70743	30942	2991	1422
K = 2, T = 48	127531	3146849	126348	3146794	861556	13094	6855
k = 31	*	-	*	-	*	*	*
Tsp, 14	607	9889	573	9855	7785	1097	631
K = 2, T = 16	4445	130021	4240	129936	88873	2884	1639
k = 14	7.15	26.47	7.24	24.09	X	X	0.07
Tsp, 19	1291	27448	1222	27379	17880	2568	1141
K = 3, T = 24	14765	572087	14327	571925	289048	6566	2984
k = 19	22.83	X	X	X	X	X	0.13
Blocks, 12	348	6135	337	6124	6123	569	534
K = 1, T = 12	1886	58969	1789	58938	59715	1886	1671
k = 12	0.1	2.18	0.1	2.2	44.44	0.12	0.09
Logistics, 18	1856	11162	1778	11084	11048	2114	836
K = 6, T = 18	13154	168263	12864	168126	168764	5577	2408
k = 18	10.77	260.9	8.51	265.32	*	3.71	0.21
Logistics, 21	2787	16360	2682	16255	16206	3109	1079
K = 7, T = 21	23003	293490	22633	293330	294078	8066	3144
k = 21	24.38	*	18.06	*	*	9.53	0.29
Build House, 23	1267	17377	1252	17362	16056	29451	1105
K = 3, T = 24	14645	353735	14333	353699	313239	293189	2532
k = 23	3.78	*	3.88	*	*	*	0.28
Tire World, 11	284	4476	280	4472	4471	1994	465
K = 1, T = 11	1480	38657	1415	38647	39429	10605	1567
k = 11	0.04	1.09	0.05	1.07	31.46	0.68	0.05

Figure 2: Empirical results on various encodings. V, C, T denote the number of variables, clauses in and the times needed to solve the encodings respectively. Times are in CPU seconds. A “*” indicates that the encoding could not be solved within 10 minutes. A “-” denotes that the encoding was too large to store. X denotes that the encoding could not be solved with Walksat with default parameters. k denotes the number of steps in the action-based encodings. TDCP and TDCN denote top-down causal pre-processed and top-down causal naive HTN encodings respectively. BUCP and BUCN denote bottom-up causal pre-processed and bottom-up causal naive HTN encodings respectively. SBH denotes the state-based HTN encoding and AC and AS denote action-based causal and action-based state-based encodings respectively.

differs from this work in at least two aspects - **(1) Nature of Knowledge** - [Kautz & Selman 98] use state constraints, e.g. once a vehicle is loaded (at time t), it should move immediately (at time $(t+1)$). The reduction schemas we used contain different types of constraints like the causal links and precedences. Certain encodings are naturally compatible with certain constraints, e.g. the state constraints and the state-based encodings, the causal links, partial order and the causal encodings. Just like adding state constraints may significantly increase the size of a causal encoding, adding causal constraints (like links and \prec) significantly increases the size of a state-based encoding, as illustrated by our results in the domains like house building and tire world. **(2) Organization of the Knowledge** - The domain specific knowledge in [Kautz & Selman 98] is not necessarily problem dependent. On the other hand, the reduction schemas generally contain constraints that must be satisfied by the solution of a specific problem in the domain. The commitment to using a reduction schema means a commitment to satisfying all of its constraints. However the constraints in [Kautz & Selman 98] are not grouped together as those in the reduction schemas and thus one can choose to add some constraints to an encoding ignoring others.

6 Conclusion

Hitherto, it was unclear if the HTN encodings of [Mali & Kambhampati 98] were easier to solve than the action-based encodings and whether they were solvable at all, since their sizes are higher than the corresponding action-based encodings and the action-based causal encodings have been shown to be hard to solve [Mali & Kambhampati 99]. We showed that the causal HTN encodings are significantly easier to solve than the causal action-based encodings, by developing a pre-processing procedure that reduced their sizes significantly. We also added constraints to the HTN encodings of [Mali & Kambhampati 98] to force their models to be minimal plans and obey the parsing criterion, to respect the user intent. We showed that the models of the causal HTN encodings indeed have these properties.

References

[Barrett & Weld 94] Anthony Barrett and Daniel Weld, Task-Decomposition

via plan parsing, Proceedings of the National Conference on Artificial Intelligence (AAAI), 1994, 1117-1122.

[**Blum & Furst 95**] Avrim Blum and Merrick Furst, Fast planning via planning graph analysis, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[**Ernst et al 97**] Michael Ernst, Todd Millstein and Daniel Weld, Automatic SAT compilation of planning problems, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997.

[**Erol 95**] Kutluhan Erol, Hierarchical task network planning: Formalization, Analysis and Implementation, Ph.D thesis, Dept. of computer science, Univ. of Maryland, College Park, 1995.

[**Estlin et al. 97**] Tara Estlin, Steve Chien and Xuemei Wang, An argument for a hybrid HTN/Operator-based approach to planning, Proceedings of the European Conference on Planning (ECP), 1997, 184-196.

[**Giunchiglia et al 98**] 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.

[**Kambhampati et al. 98**] Subbarao Kambhampati, Amol Mali & Biplav Srivastava, Hybrid planning in partially hierarchical domains, Proceedings of the National Conference on Artificial Intelligence (AAAI), 1998.

[**Kambhampati 97**] Subbarao Kambhampati, Challenges in bridging the plan synthesis paradigms, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997.

[**Kautz et al. 96**] Henry Kautz, David McAllester and Bart Selman, Encoding plans in propositional logic, Proceedings of the conference on Knowledge Representation & Reasoning (KRR), 1996.

[**Kautz & Selman 96**] Henry Kautz and Bart Selman, Pushing the envelope: Planning, Propositional logic and Stochastic search, Proceedings of the National Conference on Artificial Intelligence (AAAI), 1996.

[**Kautz & Selman 98**] Henry Kautz and Bart Selman, The role of domain-specific knowledge in the planning as satisfiability framework, Proceedings of the international conference on Artificial Intelligence Planning Systems (AIPS), 1998.

[**Mali & Kambhampati 98**] Amol D. Mali and Subbarao Kambhampati, Encoding HTN planning in propositional logic, Proceedings of the International Conference on AI Planning Systems (AIPS), 1998.

[**Mali & Kambhampati 99**] Amol D. Mali and Subbarao Kambhampati, On the utility of causal encodings, Proceedings of the National Conference on Artificial Intelligence (AAAI), 1999.

[**Tate 77**] Austin Tate, Generating project networks, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1977, 888-893.

[**Wilkins 88**] David Wilkins, Practical planning: Extending the classical AI planning paradigm, Morgan Kaufmann, 1988.