Encoding HTN Planning in Propositional Logic*

Amol D. Mali and Subbarao Kambhampati
Department of Computer Science & Engineering,
Arizona State University, Tempe, AZ 85287-5406
Email: {amol.mali, rao}@asu.edu
URL: http://rakaposhi.eas.asu.edu/yochan.html

Abstract

Casting planning problems as propositional satisfiability problems has recently been shown to be an effective way of scaling up plan synthesis. Until now, the benefits of this approach have only been utilized in primitive action-based planning models. Motivated by the conventional wisdom in the planning community about the effectiveness of hierarchical task network (HTN) planning models, in this paper we adapt the "planning as satisfiability" approach to HTN planning models. HTN planning models can be thought of as an augmentation of primitive action based planning models with a grammar of legal solutions, provided in the form of non-primitive tasks and task reduction schemas. Accordingly, we argue that any action-based encoding scheme can be generalized to handle HTN planning models. Informally, this generalization involves adding constraints to the encoding to ensure that the solutions produced by solving the encoding will conform to the grammar provided by the HTN planning model. The constraints can be added in either a "top-down" or "bottom-up" fashion, resulting in two HTN encoding schemes for each primitive action-based encoding scheme. We illustrate this process by providing three different HTN encodings. We discuss the asymptotic sizes of these encodings, as well as the complexity of finding models for them.

1 Introduction

Recent work [Kautz & Selman 96; Kautz *et al.* 96] has shown that a surprisingly effective way of attacking the classical planning problem is to pose it as a propositional satisfiability problem. Planners based on this approach have been used to solve large problems in benchmark "toy" domains with orders of magnitude reduction in synthesis time. A logical next step for the planning community would be to exploit these approaches to develop planners for complex real world domains. Unfortunately, there is one impediment to following this course. Conventional wisdom in the planning community, supported to a large extent by the fielded applications to-date, holds that most real world domains are best modeled with hierarchical task network (HTN) planning models. In contrast the "planning as satisfiability" approach has hither-to been applied only to primitive action based planning models. In this paper, we aim to overcome this mismatch by adapting the planning as satisfiability approach to HTN planning.

Adapting the satisfiability approaches to HTN planning models presents several immediate challenges. To begin with, there is some disagreement within the planning community as to the relation between the HTN models and action-based models. In this paper, we view HTN planning as an "augmentation" of the action-based planning paradigm, where the abstract tasks and their associated reduction schemas provide an implicit grammar of legal ("desired") solutions [Kambhampati *et al.* 98]. An immediate advantage of this view is that propositional encodings for HTN planning models can be developed by generalizing each of the action-based encodings in [Kautz *et al.* 96]. The basic idea is to constrain the encodings such that their satisfying models not only correspond to valid solutions to the planning problems, but also conform to the grammar specified by the reduction schemas. The constraints can be added in either a "top-down" or "bottom-up" fashion, resulting in two HTN encoding schemes for each primitive action-based encoding scheme. The "top-down" encodings are developed by introducing (disjunctively) all non-primitive tasks capable of solving the problem, and then using the task reduction schemas to add constraints to the encoding to relate the non-primitive actions to primitive actions. The "bottom-up" encodings use the task reduction schemas to ensure that the solutions for the encodings can be parsed in terms of the reductions.

We illustrate our methodology by providing three different HTN encodings. Since most implemented

This research is supported in part by NSF young investigator award (NYI) IRI-9457634 and ARPA/Rome Laboratory planning initiative grant F30602-95-C-0247.

HTN planners [Wilkins 88; Tate 77] share a lot of structure of the plan-space planners, the "causal encodings" in [Kautz *et al.* 96] provide a natural starting place for developing encodings for HTN planning. We provide both "top-down" and "bottom-up" HTN encodings based on causal encodings. However, given our view of HTN planning as just an augmentation of action-based planning with a grammar of legal solutions, there is no reason to constrain ourselves to causal encodings. To emphasize this point, we also develop a "bottom-up" encoding based on the forward state-space encoding of [Kautz *et al.* 96].

In the rest of this paper, we develop the three encodings motivated above, and analyze their asymptotic properties. Due to the elaborate nature of HTN planning models, as compared to the primitive action-based models, the specification of the encodings involves a larger number of encoding schemas as compared to the primitive action-based encodings. The paper is organized as follows: In Section 2, we review the HTN planning model and the planning as satisfiability approach. In Section 3, we explain the representation of reduction schemas that we use, and introduce the notation used in the encodings. In the following three sections (4, 5 and 6), we explain how the top-down and the bottom-up causal encodings, as well as the bottom-up forward linear HTN encodings, are generated automatically for a given problem. In each case, we analyze the size of the encoding. In Section 7, we compare the HTN encodings with the encodings based on only primitive actions and discuss ways of efficiently solving the HTN encodings. Section 8 presents the conclusions.

2 Background

HTN Planning: Traditionally, the planning problem is posed as one of finding an action sequence that will transform an agent's world from a specified initial state into a desired goal state, given only the description of the executable actions in the domain. In many realistic domains there exist human experts who are ready to share their significant planning experience with automated planners. Efficiency of a planner, as well as the acceptability of solutions produced by it, may then depend crucially on the ability to effectively use this expertise. The conventional wisdom in the planning community has been that the non-primitive actions along with action reduction schemas (such as the one shown in Figure 1) for converting them into other non-primitive and primitive actions, provide a flexible way of capturing the human expertise, as well as using it to control planning. Planners that use such action reduction schemas have come to be known as "hierarchical task network planners" [Sacerdoti 77][Wilkins 88][Erol 95]. 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 *at al.* 97].

The action reduction schemas capture the human expertise in that only those solutions that have a parse in terms of the supplied schemas are considered legal. The rich structure of the reduction schemas (see Section 3) allows the users to provide a fairly sophisticated grammar for legal solutions [Kambhampati *et al.* 98]. The 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). 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.

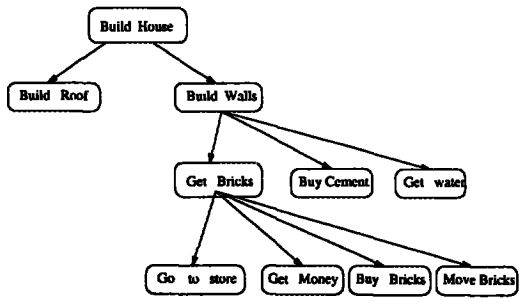

Figure 1: Hierarchical task network for building a house

Posing Planning as Satisfiability: Given a planning problem $\langle I, G, A \rangle$, where I is the initial state of the world, G specifies the desired goals and A is the set of actions that an agent can execute, [Kautz & Selman 96] show that the task of finding a k-step solution to this problem can be posed as a propositional satisfiability problem. 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 must capture various constraints required for proving that a k-length action sequence is a solution to the planning problem. Three classes of encodings have been developed, corresponding broadly to three ways of carrying out this proof – (i) forward

encodings that comprise constraints required to show that if the actions are progressed (executed) from the initial state, the goals will hold in the final state; (ii) backward encodings that comprise the constraints required to show that if the goals are regressed over the action sequence, the result will be subsumed by the initial state; (iii) "causal" encodings that comprise the constraints required to show that every top level goal as well as every precondition of the actions in the sequence are caused by the effect of some preceding action in the sequence, such that any action coming after the supporting action preserves the goal. [Kautz *et al.* 96] report encodings based on each of these ideas.[1] The contribution of the current paper can be seen as providing a methodology for adapting these three classes of encodings to HTN planning models.

3 Representation & Notation

In order to formally develop HTN encodings, we must first agree on the representation of the task reduction schemas, since these are the means by which the grammar of the legal solutions are specified. We describe the details of this representation in the following. The representation we use is consistent with that used in [Erol, 95], which in itself is a generalization of the schema representation used in [Tate, 77]. We also develop some notation that will be used in compactly specifying the encodings in Sections 4, 5 and 6.

In what follows, we assume that o_i denotes a STRIPS style primitive action. The number of *add* effects, *delete* effects and preconditions of o_i are denoted by A_i, D_i and R_i respectively. a_{ik} denotes kth add effect of o_i, d_{iz} denotes zth delete effect of o_i and $n_{iz'}$ denotes z'th precondition of o_i. p_i denotes a primitive step that is mapped to a primitive action. O denotes the set of all primitive actions from the domain.

N_i denotes a 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 a non-primitive task. C_j denotes the number of *Add* effects of N_j. E_{jp} denotes the pth *Add* effect of N_j. r_{ij} denotes jth reduction schema of N_i. Each reduction schema r_{ij} is specified in terms of the following 14 types of constraints (that capture the causal links and orderings between non-primitive tasks and primitive actions) - (i) mapping from an action symbol to a primitive action, for example, $o_1 : load(x, l_1)$, (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 a causal link where o_p is an action that has the *add* effect f and o_q is an action that has the precondition f. o_p is called the contributor and o_q is called the consumer. (vi) $? \xrightarrow{f} o_p$, (which denotes that the precondition f of the action o_p will be supplied by some action that is not known a priori, but 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) $N_p \xrightarrow{f} ?$.

Example: We will now illustrate the schema representation with a simple example domain called the "one-way rocket" domain. Here, the packets A and B that are initially on Earth (initial state) are to be transported to Moon (goal state). The domain specification consists of two non-primitive tasks: $N_1 : Transport(x, l_1, l_2)$ is the task of transporting packet x from place l_1 to place l_2. There is one reduction schema associated with it, which says that to transport the packet, one must first put it in the rocket, transport the rocket to the moon (which is a non-primitive action), and take the packet out of the rocket. Formally, the reduction schema is specified as follows:

$$\left\langle \begin{array}{c} o_1 : load(x, l_1), N_2 : achieve(at(R, l_2)), \\ o_2 : unload(x, l_2), \\ o_1 \prec N_2, N_2 \prec o_2, \\ o_1 \xrightarrow{in(x,R)} o_2, N_2 \xrightarrow{at(R,l_2)} o_2, \\ ? \xrightarrow{at(x,l_1)} o_1, o_2 \xrightarrow{at(x,l_2)} ? \end{array} \right\rangle$$

The non-primitive task N_2 is the task of getting the rocket R to location l_2. It has two reductions. The first reduction says that the condition that the rocket should be at place l_2 is already true, hence the only element of that reduction is the null action o_3. PTC denotes point truth constraint that asserts that a particular condition must be true at a particular time. This is specified as:

$$\langle o_3 : no-op, PTC : at(R, l_2)@o_3 \rangle$$

The other reduction of N_2 is the primitive action $fly(R, l_1, l_2)$, that flies the rocket R from place l_1 to place l_2.

$$\langle o_4 : fly(R, l_1, l_2) \rangle$$

Notice that, to successfully solve the one-way rocket problem, the primitive actions from the reductions of

[1] Since these three types of proofs of plan correctness have traditionally been used in the forward state-space, backward state-space and plan-space planners respectively, the encodings have also been associated with the planners. Accordingly, the first two can be seen as state-space encodings while the third can be seen as a plan-space encoding.

N_1 and N_2 have to be interleaved properly to get the plan. From now on when we refer to a task, we mean a non-primitive task and when we refer to an action, we mean a primitive action (for brevity).

Preliminaries in setting up an encoding: In an encoding, we need to have an economical way (that avoids combinatorially large number of clauses) of referring to the possible ways of satisfying the constraints from a reduction schema. $t(r_{ij})_k$ denotes a transformation of the reduction schema r_{ij} to consider the mapping $s_k = N_i$. Mapping the step s_k to N_i means choosing N_i to solve the planning problem. In this case, it is necessary to be able to carry out the task N_i by fulfilling constraints in some reduction schema of N_i. $t(r_{ij})_k$ represents a conjunction of the disjunction of all possible ways of satisfying each constraint in r_{ij} (non-primitive actions can thus be seen as "*disjunctive constraints*" on the partial plan. The process of reducing non-primitive actions can be seen as making the implicit disjunction explicit). For example, $t(r_{22})_1$ in the encoding for the previously discussed instance of the transportation problem represents all possible ways of satisfying the constraints from the second reduction schema of N_2, when the non-primitive step s_1 is mapped to N_2. A reduction schema can have more than one transformation because the same reduction schema may be needed more than once, and in that case, we will have to use different primitive steps in different occurrences of the reduction schema.

We assume that there are m non-primitive tasks, some of which are relevant to solving the planning problem. M_i denotes the maximum number of primitive actions in reduction schema of N_i. $M = max(\{M_i \mid i \in [1,m]\})$. Knowing M allows us to *a priori* compute the number of primitive steps required in setting up an HTN encoding. When s_k is mapped to N_i and the reduction schema r_{ij} is used to reduce N_i, primitive steps from $p_{a'_{ijk}}$ to $p_{a''_{ijk}}$ are mapped to primitive actions, 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}.

Our encoding schemes use ground primitive actions and ground non-primitive tasks. However it is easy to extend them to variablized primitive actions and non-primitive tasks. We capture in the encodings the scenario where both hierarchical task networks and primitive actions (that do not occur in the hierarchical task networks) are used for planning. Such a hybrid planning process is quite common since many domains are only partially hierarchized [Kambhampati *et al.* 98].

Our encoding schemes create an encoding for K non-primitive steps and an additional k' primitive steps (which are mapped to primitive actions that do not belong to any reduction schema). The reduction of non-primitive tasks requires T primitive steps in the encoding. The total number of primitive steps in the encoding are $(T+k')$, where $T = M*K$. The primitive steps are denoted by $p_1, p_2, ..., p_{T+k'}$. The goal state G is assumed to be a conjunction of literals $(a_1 \wedge a_2 \wedge ... \wedge a_h)$. F denotes the last step of a plan which has no effects and whose preconditions are same as the conditions in G. A planning problem may require less than $(T+k')$ primitive actions. Hence we use the null actions (denoted by ϕ) that have no preconditions, no effects and no reductions.

4 Top-Down Causal HTN Encoding

In this encoding, we use reduction schemas to translate tasks into less abstract ones. These translations are embedded in the encoding schemas. Hence the encoding is defined to be top-down. During these translations, we need to generate all possible ways of satisfying the constraints in the reduction schemas, to achieve soundness and completeness.

When a non-primitive step is mapped to a task, the step adds the effects of the task. We encode that if a non-primitive step is mapped to a task, there should be a way to fulfill the constraints in some reduction of that task. Consider the package transportation example from section 3. Let us create a 3-non-primitive step encoding for this problem using 3 ground tasks : transporting A to Moon, transporting B to Moon and having the rocket at Moon. Their reduction schemas contain 2, 2, and 1 actions respectively. Since each of the 3 non-primitive steps can be mapped to any of these tasks and the maximum number of primitive actions M in a reduction schema is 2, we need 6 primitive steps ($T = K * M$, as explained in section 3). We then use the schemas in Figure 2 to generate the encoding. (Note that the encoding schemas and reduction schemas have different semantics.) In this problem, we do not require any primitive action that does not occur in the reduction schemas ($k' = 0$).

Recall that this encoding is based on the causal encoding of [Kautz *et al.* 96]. Below we explain what the schemas in Fig. 2 mean. Schema 1 states that a non-primitive step can be mapped to any non-primitive task, including the null action. Schema 2 states that every literal in the goal state is added by some non-primitive step or a primitive step. Schema 3 states that a non-primitive step cannot be mapped to more than one non-primitive task. Schema 6 states that if a non-primitive step is mapped to a particular non-

primitive task, some transformation of some reduction of the non-primitive task must be true. This schema has 14 parts for handling 14 types of constraints that may occur in a reduction schema. **(i)** states that primitive steps that are mapped to primitive actions from a reduction schema inherit the preconditions and effects of those primitive actions. **(ii)** An element $N_i : task - name$ is handled by requiring some non-primitive step to be be mapped to N_i. **(iii)** states that an existence of an element of type $o_s \prec N_p$ requires the primitive step p_y mapped to o_s to precede all primitive steps p_q in some transformation of some reduction of N_p (since we do not know *a priori* which non-primitive step will be mapped to N_p and which reduction schema will be used to decompose N_p). **(iv)** is similar to **(iii)**. **(v)** states that when there is a causal link $o_p \xrightarrow{f} o_q$, there exist primitive steps p_{y_1}, p_{y_2} mapped to o_p and o_q respectively, and there exists a causal link between them. **(vi)** states that when there is a causal link $? \xrightarrow{f} o_p$ in a reduction schema, some primitive step p_z or initial state provides the precondition f to the primitive step p_y that is mapped to o_p. **(vii)** is similar to **(vi)**. **(viii)** says that when there is a constraint $N_q \xrightarrow{f} o_p$ in a reduction schema, some primitive step $p_{q'}$ from some transformation of some reduction of N_q provides the condition f to the primitive step p_y that is mapped to o_p. **(ix)** is similar to **(viii)**. **(x)** states that when there is a constraint $N_p \prec N_q$ in a reduction schema, all primitive steps p_{u_1} in some transformation of some reduction of N_p precede all primitive steps p_{u_2} in some transformation of some reduction of N_q. **(xi)** states that when there is a constraint $N_p \xrightarrow{f} N_q$, some primitive step p_{u_1} in some transformation of some reduction of N_p provides the condition f to some primitive step p_{u_2} in some transformation of some reduction of N_q. **(xii)** states that when there is a constraint $o_p \prec o_q$, the primitive step p_{y_1} mapped to o_p precedes the primitive step p_{y_2} mapped to o_q. **(xiii)** states that when there is a constraint $? \xrightarrow{f} N_q$ in a reduction schema, some primitive step p_y or the initial state provides the condition f to some primitive step p_t in some transformation of some reduction of N_q. **(xiv)** states that when there is a constraint $N_p \xrightarrow{f} ?$ in a reduction schema, some primitive step p_y or the goal step F gets the condition f from some primitive step p_t in some transformation of some reduction of N_p. For each causal link $p_i \xrightarrow{f} p_j$ occurring in the encoding, we also say that $((p_i \xrightarrow{f} p_j) \Rightarrow (Adds(p_i, f) \wedge Needs(p_j, f) \wedge (p_i \prec p_j)))$.

Schema 7 says that if constraints in some transformation of some reduction schema of a non-primitive task hold, then some non-primitive step must have been mapped to that particular non-primitive task. This schema is used to rule out unintended models. Schema 8 says that if a non-primitive step is mapped to a non-primitive task, it adds the *add* effects of that task and vice versa. When a non-primitive step is mapped to a non-primitive task, a transformation of some reduction of that task holds. However, transformation of only one reduction should hold, since the mapping $(s_i = N_j)$ demands only one occurrence of one reduction schema of N_j. Schema 9 states such mutually exclusive transformations. Formal versions of the encoding schemas 4, 5, 10 ... 20 are not shown in Fig. 2 because these schemas are similar to the schemas for causal encoding of [Kautz *et al.* 96]. These schemas state the transitivity on the orderings of primitive steps, the reordering of steps necessary for resolving threats and the need to have a contributor for each precondition of each primitive step.

The top down HTN encoding contains $O(K * m * d * Z + (T + k')^3 * \mid \Gamma \mid)$ clauses and $O((T + k')^2 * \mid \Gamma \mid)$ variables where $d = max(\{d_i \mid i \in [1, m]\})$ and Γ is the set of preconditions of all primitive actions. Z is the maximum number of clauses that get added to the encoding, when all possible ways of satisfying all the constraints in reduction schema of a task are enumerated (Schema 6 in Fig. 2 contributes these clauses).

5 Bottom-Up Causal HTN Encoding

The notions of non-primitive steps and tasks are absent in the bottom-up encoding. It uses only primitive steps that are mapped to actions. However, we do use the constraints from reduction schemas. In setting up a bottom-up encoding, it is not wise to randomly choose the number of primitive steps, e.g. in the one way rocket domain, at least 3 primitive steps (load(..), fly(..) and unload(..)) are required for the transportation task, and generating an encoding for lesser steps fails. Hence we form the bottom up encoding with a certain number of primitive steps, assuming that certain number of reduction schemas may be required certain number of times. We also add k' extra primitive steps, since a planning problem may need primitive actions that do not occur in a reduction schema. We specify that one or more transformations of one or more reduction schemas must hold. By transformation of a reduction schema, we mean a conjunction of all possible ways of satisfying the constraints from that reduction schema. The bottom up encoding contains $(T + k')$ primitive steps, where $T = K * M$ as in the top-down encoding.

Important schemas for generating this encoding are shown in Fig. 3. Schema 1 states that every condition

1. $\wedge_{i=1}^{K}(\vee_{j=1}^{m}(s_i = N_j) \vee (s_i = \phi))$

2. $\wedge_{i=1}^{h}((\vee_{j=1}^{K} Adds(s_j, a_i)) \vee (\vee_{k=T+1}^{T+k'} Adds(p_k, a_i)))$

3. $\wedge_{i=1}^{K} \wedge_{j=1}^{m} \wedge_{p=1,p\neq j}^{m}(\neg(s_i = N_j \wedge s_i = N_p))$

$\wedge_{i=1}^{K} \wedge_{j=1}^{m} (\neg(s_i = N_j \wedge s_i = \phi))$

6. $\wedge_{i=1}^{K} \wedge_{j=1}^{m} ((s_i = N_j) \Rightarrow (\vee_{s=1}^{d_j}(t(r_{js})_i)))$

(The following subschemas are used to compute $t(r_{ij})_k$).

(i) o_j : *action − name.* $(p_i = o_j) \wedge (\wedge_{k=1}^{A_j} Adds(p_i, a_{jk})) \wedge$

$(\wedge_{z=1}^{D_j} Dels(p_i, d_{jz})) \wedge (\wedge_{z'=1}^{R_j} Needs(p_i, n_{jz'}))$

(iii) $(o_s \prec N_p)$ $\quad \vee_{w=1,w\neq k}^{K}(s_w = N_p)$,

$\wedge_{u=1}^{d_p} \wedge_{w=1,w\neq k}^{K} (t(r_{pu})_w \Rightarrow (\wedge_{q=a'_{puw}}^{a''_{puw}}(p_y \prec p_q)))$

(iv) $(N_p \prec o_s)$ $\quad \vee_{w=1,w\neq k}^{K}(s_w = N_p)$,

$\wedge_{u=1}^{d_p} \wedge_{w=1,w\neq k}^{K} (t(r_{pu})_w \Rightarrow (\wedge_{q=a'_{puw}}^{a''_{puw}}(p_q \prec p_y)))$

(v) $(o_p \xrightarrow{f} o_q)$ $\quad (p_{y_1} \xrightarrow{f} p_{y_2})$

(vi) $(? \xrightarrow{f} o_p)$ $\quad (\vee_{z=1,z\neq y}^{T+k'}(p_z \xrightarrow{f} p_y) \vee (I \xrightarrow{f} p_y))$

(vii) $(o_q \xrightarrow{f} ?)$ $\quad (\vee_{z=1,z\neq y}^{T+k'}(p_y \xrightarrow{f} p_z) \vee (p_y \xrightarrow{f} F))$

(viii) $(N_q \xrightarrow{f} o_p)$ $\quad \vee_{w=1,w\neq k}^{K}(s_w = N_q)$,

$\wedge_{u=1}^{d_q} \wedge_{w=1,w\neq k}^{K} (t(r_{qu})_w \Rightarrow (\vee_{q'=a'_{quw}}^{a''_{quw}}(p_{q'} \xrightarrow{f} p_y)))$

(ix) $(o_p \xrightarrow{f} N_q)$ $\quad \vee_{w=1,w\neq k}^{K}(s_w = N_q)$,

$\wedge_{u=1}^{d_q} \wedge_{w=1,w\neq k}^{K} (t(r_{qu})_w \Rightarrow (\vee_{t=a'_{quw}}^{a''_{quw}}(p_y \xrightarrow{f} p_t)))$

(x) $(N_p \prec N_q)$

$\vee_{w_1=1,w_1\neq k}^{K} \vee_{w_2=1,w_2\neq k,w_2\neq w_1}^{K} (s_{w_1} = N_p \wedge s_{w_2} = N_q)$,

$\wedge_{k_1=1}^{d_p} \wedge_{k_2=1}^{d_q} ((t(r_{pk_1})_{w_1} \wedge t(r_{qk_2})_{w_2}) \Rightarrow$
$(\wedge_{u_1=a'_{pk_1w_1}}^{a''_{pk_1w_1}} \wedge_{u_2=a'_{qk_2w_2}}^{a''_{qk_2w_2}} (p_{u_1} \prec p_{u_2})))$

(xi) $(N_p \xrightarrow{f} N_q)$

$\vee_{w_1=1,w_1\neq k}^{K} \vee_{w_2=1,w_2\neq w_1,w_2\neq k}^{K} (s_{w_1} = N_p \wedge s_{w_2} = N_q)$,

$\wedge_{k_1=1}^{d_p} \wedge_{k_2=1}^{d_q} ((t(r_{pk_1})_{w_1} \wedge t(r_{qk_2})_{w_2}) \Rightarrow$
$(\vee_{u_1=a'_{pk_1w_1}}^{a''_{pk_1w_1}} (\vee_{u_2=a'_{qk_2w_2}}^{a''_{qk_2w_2}} (p_{u_1} \xrightarrow{f} p_{u_2}))))$

(xii) $(o_p \prec o_q)$ $\quad (p_{y_1} \prec p_{y_2})$

(xiii) $(? \xrightarrow{f} N_q)$ $\quad \vee_{w=1,w\neq k}^{K}(s_w = N_q)$,

$\wedge_{u=1}^{d_q} \wedge_{z=1,z\neq k}^{K} (t(r_{qu})_z \Rightarrow (\vee_{y=1,y\neq t}^{T+k'} \vee_{t=a'_{quz}}^{a''_{quz}} (p_y \xrightarrow{f} p_t) \vee (I \xrightarrow{f} p_t)))$

(xiv) $(N_p \xrightarrow{f} ?)$ $\quad \vee_{w=1,w\neq k}^{K}(s_w = N_p)$,

$\wedge_{u=1}^{d_p} \wedge_{z=1,z\neq k}^{K} (t(r_{pu})_z \Rightarrow (\vee_{t=a'_{puz}}^{a''_{puz}} \vee_{y=1,y\neq t}^{T+k'} (p_t \xrightarrow{f} p_y) \vee \vee_{t=a'_{puz}}^{a''_{puz}} (p_t \xrightarrow{f} F)))$

7. $\wedge_{i=1}^{K} \wedge_{j=1}^{m} ((\vee_{s=1}^{d_j}(t(r_{js})_i)) \Rightarrow (s_i = N_j))$

8. $\wedge_{i=1}^{K} \wedge_{j=1}^{m} ((s_i = N_j) \Rightarrow (\wedge_{p=1}^{C_j} Adds(s_i, E_{jp})))$

$\wedge_{i=1}^{K} \wedge_{j=1}^{m} \wedge_{p=1}^{C_j} (Adds(s_i, E_{jp}) \Rightarrow \vee_{s=z_1}^{z_2}(s_i = N_s))$

9. $\wedge_{q=1}^{K} \wedge_{i=1}^{m} \wedge_{e=1}^{m} \wedge_{j=1}^{d_i} \wedge_{f=1,\neg(i=e\wedge j=f)}^{d_e} \neg(t(r_{ij})_q \wedge t_(r_{ef})_q)$

Figure 2: Schemas for Top-Down causal HTN Encoding

1. $(\wedge_{i=1}^{h}(\vee_{j=1}^{T+k'} Adds(p_j, a_i)))$

5. $\vee_{i=1}^{K} \vee_{j=1}^{m} \vee_{s=1}^{d_j}(t(r_{js})_i)$

6. (used to compute $t(r_{ij})_k$).

(iii) $(o_s \prec N_p)$ $\quad \vee_{u=1}^{d_p} \vee_{w=1,w\neq k}^{K} (\wedge_{q=a'_{puw}}^{a''_{puw}}(p_y \prec p_q))$

(iv) $(N_p \prec o_s)$ $\quad \vee_{u=1}^{d_p} \vee_{w=1,w\neq k}^{K} (\wedge_{q=a'_{puw}}^{a''_{puw}}(p_q \prec p_y))$

(ix) $(o_p \xrightarrow{f} N_q)$ $\quad \vee_{u=1}^{d_q} \vee_{w=1,w\neq k}^{K} \vee_{t=a'_{quw}}^{a''_{quw}} (p_y \xrightarrow{f} p_t)$

(x) $(N_p \prec N_q)$

$\vee_{k_1=1}^{d_p} \vee_{k_2=1}^{d_q} \vee_{w_1=1,w_1\neq k}^{K} \vee_{w_2=1,w_2\neq k,w_1\neq w_2}^{K}$
$(\wedge_{u_1=a'_{pk_1w_1}}^{a''_{pk_1w_1}} \wedge_{u_2=a'_{qk_2w_2}}^{a''_{qk_2w_2}} (p_{u_1} \prec p_{u_2}))$

(xi) $(N_p \xrightarrow{f} N_q)$

$\vee_{k_1=1}^{d_p} \vee_{k_2=1}^{d_q} \vee_{w_1=1,w_1\neq k}^{K} \vee_{w_2=1,w_2\neq w_1,w_2\neq k}^{K}$
$(\vee_{u_1=a'_{pk_1w_1}}^{a''_{pk_1w_1}} \vee_{u_2=a'_{qk_2w_2}}^{a''_{qk_2w_2}} (p_{u_1} \xrightarrow{f} p_{u_2}))$

(xiii) $(? \xrightarrow{f} N_q)$

$\vee_{u=1}^{d_q} \vee_{z=1,z\neq k}^{K} (\vee_{y=1,y\neq t}^{T+k'} \vee_{t=a'_{quz}}^{a''_{quz}} (p_y \xrightarrow{f} p_t)) \vee$
$(\vee_{t=a'_{quz}}^{a''_{quz}} (I \xrightarrow{f} p_t))$

Figure 3. Schemas for Bottom-Up Causal HTN Encoding

in the goal state should be added by some primitive step. Schema 5 states that the disjunction of all transformations of all reduction schemas must be true. This is required for ruling out plans that do not have a parse in terms of any reduction schema. Note that nowhere in the bottom up encoding schemas do we use task (N_i) and non-primitive step (s_i) symbols. Despite the use of $t(r_{ij})_k$ (defined in sec. 3), the bottom up causal encoding remains different from the top-down encoding, since it lacks the heterogeneous structure arising from an explicit distinction between primitive and non-primitive tasks and steps. The bottom-up causal encoding contains $O(K*m*d*Z+(T+k')^3* \mid \Gamma \mid)$ clauses and $O((T+k')^2* \mid \Gamma \mid)$ variables. Some schemas of bottom up causal encoding are not shown in Figure 3 because they also occur in the top down encoding scheme.

6 Bottom-up forward linear HTN Encoding

We now change tracks and present an HTN encoding based on forward linear encoding of [Kautz & Selman 96]. Specifically, we constrain their linear forward encoding with the constraints from the reduction schemas (in a bottom-up fashion, without referring to non-primitive tasks). Similar development can also be done for their parallel encodings, which allow multiple actions at each time step.

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. In addition to the schemas from the linear encoding of [Kautz *et al.* 96] (which we do not state here), we need a schema that tells how the constraints from a hierarchical task network are to be encoded. We discuss below how we handle the constraints from reduction schema r_{ij}. **(i)** Consider elements of type $o_i : action - name$. There should be a time step $t \in [0, T+k'-1]$ at which o_i occurs. **(ii)** Each element of type $N_j : task - name$ in a reduction schema indicates that the constraints from some reduction of N_j must be satisfied. **(iii)** The constraint $o_s \prec N_p$ is handled by stating that all primitive actions in some reduction of N_p should occur at time steps that succeed the time step at which o_s occurs. In the notation,

$$\bigvee_{j=1}^{d_p} \bigwedge_{o_q \in r_{pj}} \Big(\bigvee_{i=1}^{T+k'-1} (o_q(i) \bigwedge \big(\bigvee_{k=0}^{i-1} o_s(k) \big) \Big) \Big)$$

(iv) The constraint $N_p \prec o_s$ is handled by saying that all primitive actions in some reduction of N_p should occur at time steps that precede the time step at which o_s occurs. **(v)** The constraint $o_p \xrightarrow{f} o_q$ is handled by saying that there should be time steps i, j such that i precedes j and o_p, o_q occur at i, j respectively and the proposition f is true at all time steps from $(i+1)$ to j. **(vi)** The constraint $? \xrightarrow{f} o_p$ is handled by saying that a primitive action which adds f and occurs at time step i or initial state provides f to o_p, which occurs at time j succeeding i and f is true from time $(i+1)$ to j. **(vii)** The constraint $o_q \xrightarrow{f} ?$ is handled by stating that o_q occurs at time i and some primitive action which needs f (occurring at time $j > i$) or the goal gets f and f is true over the interval $[i+1, j]$. **(viii)** The constraint $N_q \xrightarrow{f} o_p$ is handled by stating that some primitive action in some reduction of N_q which adds f occurs at time i, o_p occurs at $j > i$ and the proposition f is true over the interval $[i+1, j]$. **(ix)** The constraints of type $o_p \xrightarrow{f} N_q$ are handled similar to the constraint **(viii)**, with a minor variation. **(x)** To handle the constraint $N_p \prec N_q$, we state that each primitive action in some reduction of N_p occurs at a time step before each primitive action in some reduction of N_q. **(xi)** To handle the constraint $N_p \xrightarrow{f} N_q$, we say that some primitive action in some reduction of N_p which adds f occurs at time i, some primitive action in some reduction of N_q which needs f occurs at time $j > i$ and the proposition f is true over the time interval $[i+1, j]$. **(xii)** The constraint $o_p \prec o_q$ is handled by stating that o_p occurs at a time step before o_q. **(xiii)** $? \xrightarrow{f} N_q$ **(xiv)** $N_p \xrightarrow{f} ?$ are handled similar to **(xi)**, however, the contributor is unknown in **(xiii)** (the contributor can be initial state or any primitive action which has f in the list of *Add* effects) and the consumer in **(xiv)** is unknown (it can be any primitive action that needs f or the goal itself).

The total number of clauses and variables in the linear encoding are $O(d*m*3^{((T+k')^2*|O|*b)} * ((T+k'+1)!)^b)$ and $O((T+k')* \mid O \mid +(T+k')* \mid O \mid * \mid F' \mid)$ respectively, where F' is the set of all propositions from the preconditions and effects of the domain actions, $d = max(\{d_i \mid i \in [1,m]\})$, and $b = max(\{\sum_{j=1}^{d_q} b_{qj} \mid q \in [1,m]\})$. The number of clauses can be reduced to $O(b'*d^2*m*(T+k')^3* \mid O \mid)$ by introducing additional variables. The number of variables then becomes $O(b'*d^2*m*(T+k')^2* \mid O \mid)$, $b' = max(\{b_{ij} \mid i \in [1,m], j \in [1,d_i]\})$. These expressions also show that the linear encoding for HTN planning competes well with the causal encodings for HTN planning (top-down and bottom-up) that have $O(K*m*d*Z+(T+k')^3* \mid \Gamma \mid)$ clauses and $O((T+k')^2* \mid \Gamma \mid)$ variables.

7 Discussion

Soundness and Completeness of HTN Encodings

Since the primary motivation behind using HTNs is to respect the preferences of users, the completeness of HTN planners cannot be judged with respect to the set of all action sequences, but rather only with respect to those sequences that can be parsed by the schemas. Our top-down encoding is set up in such a way that it will have a model if and only if a solution can be found by choosing and decomposing m or less tasks and choosing k' or less extra primitive actions and resolving the interactions among them. If this is not the case, the encoding will not have a model. The top-down encoding contains all tasks, actions, interactions and ways of resolving these. The top-down encoding is sound and complete.

The bottom-up causal encoding requires that the constraints from at least one reduction schema need to be respected by any model. Each transformation $t(r_{ij})_k$ of each reduction of a task maps certain steps to actions. Hence each of the T steps cannot be mapped to any arbitrary action, but rather to only one of the actions contained in the reduction schemas. The encoding also states that each mapping of certain steps to primitive actions implies the disjunction of all transformations which contain that mapping. The reason is that if a group of primitive actions is relevant to the planning problem, one or more reduction schemas in which this group appears must be relevant to the problem as well, and in that case there should be a way of satisfying all constraints in those reduction schemas. This disallows solutions that do not have a parse in terms of the reduction schemas, and along with other constraints in the encoding, ensures that a model can be found if and only if a plan of $(T + k')$ or less steps exists. Similar arguments can be used to establish the soundness and completeness of the bottom-up forward linear HTN encoding.

Comparison with non-HTN encodings

Planning with actions can be viewed as a special case of HTN planning where each reduction schema contains a primitive action and nothing else. Hence we can compare the encodings of HTN planning with the encodings of planning without abstractions. If we decide to create an encoding with $(T+k')$ primitive steps, without using the information provided by the reduction schemas, the causal encodings for HTN planning shrink to the causal encoding provided by [Kautz *et al.* 96] with $O((T+k')^3 * \mid \Gamma \mid)$ clauses and $O((T+k')^2 * \mid \Gamma \mid)$ variables. We observe that HTN causal encodings have more clauses and variables than the causal encodings without abstractions. A comparison also shows that the linear HTN encoding has more clauses and variables than the linear encodings without abstract actions.

The comparison above raises an interesting question about the practical utility of HTN encodings. The conventional wisdom in the SAT community is that the complexity of solving a SAT encoding depends on the number of variables in the encoding. This, in juxtaposition with the fact that HTN encodings are bigger than the corresponding encodings for action-based planning, seems to imply that HTN approaches may not be superior when we pose planning as a satisfiability problem. In the worst case, this may well be true as the worst case complexity of HTN planning is indeed higher than that of normal action-based planning [Erol 95]. In practice however, we speculate that the HTN encodings can still out perform the action-based encodings for two reasons. First, recent work by [Kautz & Selman 98] shows that when domain specific knowledge about desirable solutions is added to linear action-based encodings (thus increasing their size), the complexity of testing the satisfiability of the resulting encodings *goes down*. This happens because the added domain knowledge interacts with the base-level encodings to support a significant amount of simplification through unit propagation and other techniques. Consequently, the final encoding is significantly reduced in size. Since, as we argued, HTN encodings can be seen as merely providing a grammar of the desirable solutions on the top of action-based encodings, similar simplifications may occur on HTN encodings too.

Secondly, even after simplification, we can use the constraints from the reduction schemas to "guide" the planning process. Indeed, since the early days of HTN planning [Sacerdoti 73], it has been known that the success of HTN planners heavily depends upon producing good plans at the abstract level. Doing this is complicated in our case by the fact that since the encoding is a purely declarative representation, there is no way for a solver to know the hierarchical control implicit in the encoding. One way out is to assign weights to the clauses in an HTN encoding to enforce hierarchical control. Then the solution should be found by satisfying the clauses in a hierarchical order. Since all three encodings do refer to the constraints in the reduction schemas, the weighting scheme can be used in all of them. [Jiang *et al.* 95] discuss an extension of a local search algorithm for handling weighted propositional satisfiability and show that weighted declarative representation and stochastic search have the potential to solve problems of practical interest.

Optimizing HTN encodings

We have found several ways of reducing the size of the encodings, two of which are mentioned below.

1. The number of ground reduction schemas tends to be combinatorially large. Pre-processing technique like operator graphs [Peot & Smith 96] can be used to prune irrelevant schemas.

2. The causal encoding of planning with primitive actions [Kautz *et al.* 96] contains all potential causal links. If we have reduction schemas which contain all causal links relevant to the class of problems being solved, the generation of causal links in brute force style is no longer necessary.

8 Conclusion

In this paper we adapted the planning as satisfiability framework to HTN planning models. We have argued that HTN encodings can be generated by constraining the primitive action-based encodings to ensure that all the models of the encodings also conform to the grammar of legal solutions specified by the non-primitive tasks, and their reduction schemas. This conformance can be enforced in either a top-down or a bottom-up fashion. We illustrated this methodology by providing three encodings–two based on the causal planning, and one based on the forward linear planning. We have analyzed the asymptotic sizes of these three encodings, explained their significance, and compared them with non-HTN encodings. We have also discussed the issues involved in efficiently solving the HTN encodings. We are currently working on empirically evaluating the effectiveness of these encodings. As of this writing, we have generated HTN encodings for house building, tire change and transportation logistics domains.

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.

[**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.

[**Jiang *et al.* 95**] Yuejun Jiang, Henry Kautz and Bart Selman, Solving problems with hard and soft constraints using stochastic algorithm for MAX-SAT, First international joint workshop on artificial intelligence and operations research, 1995.

[**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.

[**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.

[**Peot & Smith 96**] Mark Peot and David Smith, Suspending recursion in causal link planning, Proceedings of the international conference on Artificial Intelligence Planning Systems (AIPS), 1996.

[**Sacerdoti 73**] Sacerdoti E. D., Planning in a hierarchy of abstraction spaces, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1973, 412-422.

[**Sacerdoti 77**] Sacerdoti E. D., A structure for plans and behavior, Elsevier-North Holland, 1977.

[**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.