Abductive Completion of Plan Sketches

Karen L. Myers
AI Center, SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
myers@ai.sri.com

Abstract

Most work on AI planning has focused on the development of fully automated methods for generating plans that satisfy user-specified goals. However, users in many domains want the ability to influence the nature of the solutions that are generated. With the objective of fostering increased user participation in the planning process, this paper presents an HTN-based framework for the abductive completion of *plan sketches*. Within this framework, user-supplied outlines of partial plans (possibly spanning multiple abstraction levels) are interpreted and completed. The processing of plan sketches employs an initial abductive plan recognition phase to formulate candidate sets of intended user goals, followed by a plan refinement stage that generates *sketch-compliant* final plans for those goals. A prototype sketch-based planner based on this approach has been implemented and applied to a crisis action planning domain.

Introduction[1]

Most work on AI planning has focused on the development of fully automated techniques for generating plans that satisfy high-level user goals. These methods take a description of the domain, a problem phrased as a partially ordered set of goals, and produce a solution consisting of a set of actions that, when executed in the initial state, guarantee satisfaction of the specified goals. This approach to planning makes no provision for users to participate in the planning process.

For many applications, users are reluctant to relinquish all control to an automated system. Rather, they want to participate in the plan development process, both to guide the system toward preferred solutions, and as a means of understanding the approach taken. For example, consider planning a trip. Ideally, a traveler would like to sketch his or her itinerary, and then have an automated system fill in the details. Similarly, military campaign plans generally reflect the biases and preferences of the commanders in charge, despite the fixed doctrine that underlies much of the planning process (Thaler & Shlapak 1995).

Here, we present a framework for transforming user-supplied *plan sketches* for a given problem into complete and correct plans that comply with the user's recommendations. We refer to this process as *sketch completion*. Our notion of a plan sketch consists of a collection of goals and actions to be included in the solution to a planning problem, possibly spanning multiple abstraction levels. Plan sketches may be partial in that they contain only fragments of the overall plan rather than an end-to-end solution. They may also be partial in that goals and tasks are incompletely specified. The challenge is to fill those gaps and further refine the sketch to a complete solution to the original problem.

We adopt a two-phase approach to sketch completion. An initial *interpretation* phase generates explanations for the individual components in the sketch, using a form of abductive plan recognition to link the components to candidate high-level goals. The ensuing *refinement* phase expands selected sets of candidate high-level goals to produce full plans, using the explanation structures from the interpretation phase to ensure that the result is faithful to the original sketch.

Planning Model

We employ a Hierarchical Task Network (HTN) model of planning, based loosely on that in (Erol, Hendler, & Nau 1994).

The cornerstones of HTN planning are *task networks* and *operator schemas*. Informally, a task network is a partially ordered set of tasks (goals and actions) to be achieved, along with conditions on the world state before and after tasks are executed. Operator schemas specify methods for reducing an individual task to some new set of subtasks, under appropriate conditions. HTN planning consists of taking a description of an initial world state, an initial task net-

[1] Copyright © 1997, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

work, and a set of operator schemas for task refinement, and then repeatedly applying operator schemas until no further reductions are possible. Each such refinement may contribute additional task-ordering and world constraints to the successive task networks. The terms *task network* and *plan* are used interchangeably in this document.

Formally, we define a task network $\tau = \langle T, L, W \rangle$, where T is a set of tasks, L is a set of ordering constraints on tasks, and W is a set of world constraints. An HTN planning problem $\mathcal{P} = \langle \mathcal{O}, \tau_0, S_0 \rangle$ is modeled as a set of operator schemas $\mathcal{O}$, an initial task network τ_0, and a set of propositions S_0 denoting the initial world state. An operator schema O is characterized by its purpose $Purpose(O)$ (*i.e.*, the goals to which it can be applied), the preconditions for applying the schema $Preconds(O)$, and the task network $Tasks(O)$ to which a goal matching the purpose can be reduced by applying the schema. The tasks, constraints, and goals in the task networks and operator schemas are defined using a first-order language $\mathcal{L}$.

Tasks can be either *primitive* or *nonprimitive*, with the former having no possible refinements. The term *goal* is used interchangeably with *nonprimitive task*; similarly, *action* is used interchangeably with *primitive task*. We define the *toplevel* tasks, denoted by $T^{\uparrow}$, to be those nonprimitive tasks that appear in the purpose of some operator schema but are not subtasks in any operator schemas as A solution to an HTN problem is a refinement of the original task network that contains only primitive tasks, provided that all constraints in the task network can be resolved.

We define the *plan refinement structure* for a plan (similar to the *hierarchical task network* for a plan in (Kambhampati & Hendler 1992)) to be $\pi = \langle \mathcal{P}, \mathcal{N}, \mathcal{D} \rangle$ where $\mathcal{P}$ is the set of task networks produced, $\mathcal{N}$ is the set of nodes in any of the task networks, and $\mathcal{D}$ defines a directed acyclic graph of the refinement relations from a node to each of its descendants. $Desc(n)$ designates a node and its descendants.

Each node in a plan refinement structure has attributes defined by its associated task network. Key attributes for sketch completion include the task for the node $Task(n)$, the operator schema that has been used to expand that node $OprSchema(n)$, and the bindings for the expansion $\sigma(n)$. The notation p^{σ} denotes the instantiation of p in accord with the binding list σ, while $\sigma_1 \bigcup \ldots \bigcup \sigma_n$ denotes the composition of binding lists.

Throughout, we assume that the operator schemas are *nonrecursive*, that is, no plan refinement structures can be generated that contain nodes n and $n' \in Desc(n)$ such that $OprSchema(n) = OprSchema(n')$.

Plan Sketches

A plan sketch is a prescription of goals and actions to be included in the solution (*e.g.*, plan refinement structure) to an overall planning problem. These goals and actions can be stated over a range of abstraction levels. (In practice, we would expect a mixture of mid-range, and possibly a few low-level tasks in the typical plan sketch.) Tasks need not be specified fully: nonground goals and actions are allowed. Variables within a plan sketch are assumed to have existential interpretation, and a given variable may appear in multiple tasks, thus implicitly providing an intertask constraint.

Definition 1 (Plan Sketch) *A* plan sketch *is a set of ground and nonground tasks defined using $\mathcal{L}$.*

We refer to the tasks in a plan sketch as *anchors*.

One desired property for sketch completions is *compliance*, which ensures that elements of the sketch are reflected in the final plan.

Definition 2 (Plan Sketch Compliance) *A plan refinement structure $\pi = \langle \mathcal{P}, \mathcal{N}, \mathcal{D} \rangle$ is* compliant *with a plan sketch $\mathcal{S}$ iff there is a substitution β such that for every anchor task $\mathrm{A} \in \mathcal{S}$, there is some node $n \in \mathcal{N}$ such that $\mathrm{Task}(n)^{\sigma(n)} = \mathrm{A}^{\beta}$.*

Compliance requires that there be a consistent instantiation of the plan sketch that maps to a subset of task nodes in the final plan refinement structure.

In certain circumstances, it may be appropriate to impose requirements on sketch completions that go beyond basic compliance. For example, *minimality* would limit completions to contain only tasks that are required to cover the initial sketch. *Anchor independence* would force individual anchors to be treated as distinct components in the plan by precluding plan refinement structures in which nodes corresponding to anchors are related by task refinement links. In this paper, we consider only the basic notion of compliance.

Abductive Model of Intended Goals

Given our definition of a plan sketch as an abstract, partial outline of an overall intended plan, sketch completion requires the identification of the *intended goals* that the user is trying to solve. These goals will be drawn from a designated *goal space* $\mathcal{G}$. The goal space could be explicitly defined by the user; alternatively, the toplevel tasks $T^{\uparrow}$ provide a natural definition for $\mathcal{G}$.

We employ an abductive model to identify *candidate* intended goals from $\mathcal{G}$ for a plan sketch, building on the concept of *abductive chains*.

Definition 3 (Abductive Chains) *The* abductive chains *for a task* A *is the set of labeled linear graphs*

$$\mathrm{A} \xrightarrow{O_n,\sigma_n} G_n \xrightarrow{O_{n-1},\sigma_{n-1}} G_{n-1} \xrightarrow{O_{n-2},\sigma_{n-2}} \dots \xrightarrow{O_1,\sigma_1} G_1$$

where $G_1 \in \mathcal{G}$, and O_j is an operator schema with purpose G_j and a subtask T such that σ_j is a most-general unifier of T and $(G_{j+1})^{\beta}$, for $\beta = \bigcup_{n \geq i \geq j} \sigma_i$ and $\mathrm{A} = G_{n+1}$.

An abductive chain encodes an abstraction path from an anchor to a goal in $\mathcal{G}$, through the operator schemas. As such, it identifies both a goal that the user may be trying to solve and a kind of *explanation* for the anchor relative to that goal. There can be several such goals, with multiple chains to them. Given the assumption of nonrecursiveness for the operator schemas, there is a finite number of finite-length abductive chains for any task.

We define the *candidate goals* for a sketch to be the tasks that appear as the terminal node of an abductive chain for some anchor task in the sketch.

Example 1 Figure 1 presents a set of operator schemas for a sample domain to be used throughout the paper. To simplify the presentation, tasks are propositions, and schemas contain only a purpose and subtasks (preconditions and ordering constraints amongs subtasks are excluded). For each operator schema, the proposition to the left of $\rightarrow$ is the purpose, and the propositions to the right are the schema subtasks.

Figure 2 presents the *refinability graph* for this domain, which shows all possible applications of the operator schemas. The graph contains an edge labeled O from a task T_1 to a task T_2 iff operator schema O can be applied to refine task T_1 to task T_2. As can be seen, this domain yields two toplevel tasks, A and B.

For this domain, we define $\mathcal{G} = T^{\uparrow}$. It is straightforward to verify that task V has the abductive chains

$V \xrightarrow{O_{13}} U \xrightarrow{O_4} C \xrightarrow{O_1} B$
$V \xrightarrow{O_8} K \xrightarrow{O_3} C \xrightarrow{O_1} B$
$V \xrightarrow{O_6} D \xrightarrow{O_9} K \xrightarrow{O_3} C \xrightarrow{O_1} B$
$V \xrightarrow{O_6} D \xrightarrow{O_0} A$

while task P has the two abductive chains

$P \xrightarrow{O_5} C \xrightarrow{O_1} B$
$P \xrightarrow{O_{10}} M \xrightarrow{O_3} C \xrightarrow{O_1} B.$

The candidate goals for sketch $\{P, V\}$ are $\{A, B\}$.

In general, the user will not be attempting to satisfy every candidate goal. For instance, many of the high-level goals in a mobile robot domain will require movement between locations. Thus, a user-supplied plan sketch containing a movement task would yield a

O_0:	A	$\rightarrow$	D, E
O_1:	B	$\rightarrow$	C
O_2:	B	$\rightarrow$	Y, Z
O_3:	C	$\rightarrow$	K, L, M
O_4:	C	$\rightarrow$	U, T
O_5:	C	$\rightarrow$	P, Z
O_6:	D	$\rightarrow$	F, V
O_7:	E	$\rightarrow$	H
O_8:	K	$\rightarrow$	V, J
O_9:	K	$\rightarrow$	D
O_{10}:	M	$\rightarrow$	P, Q
O_{11}:	P	$\rightarrow$	W
O_{12}:	T	$\rightarrow$	R
O_{13}:	U	$\rightarrow$	V

Figure 1: Operator Schemas for a Sample Domain

large set of candidate goals, most of which would not be intended by the user.

Work on plan recognition has identified several types of *bias* to select subsets of the candidate goals as the *intended goals* of the user. *Goal minimization* identifies minimal subsets (by set inclusion) of the candidate goals that cover all anchors (Kautz 1990). *Referentiality* filters the candidate goals based on inclusion in a predefined *reference plan* that is serves as an ideal solution to the problem at hand (Gertner & Webber 1996).

Within the context of interpreting plan sketches, additional biases are possible. In certain situations, the user may explicitly declare his intended goals. For example, a user may inform a travel planning system that he is planning a vacation to Europe. Alternatively, a weighted matching to a reference plan could be used, in which certain critical subgoals must be covered; this bias would enable more robust interpretation in the face of plan sketches that could contain errors. In general, biases should be selected in a situation-dependent manner. Here, we make no commitment to a particular bias, instead assuming the availability of some selection function *IntendedGoals*$(\{G_1, \dots G_n\})$ to determine possible sets of intended goals from the candidates. We require only that this function *cover* the anchors, meaning that any goal set it produces contains the terminal node of some abductive chain for each anchor.

Plan Sketch Completion

This section outlines our two-phase algorithm for the abductive completion of plan sketches. First, we describe *plan skeletons*, which encode information about possible intended plans for a sketch. Next, we present an algorithm for task refinement that uses plan skele-

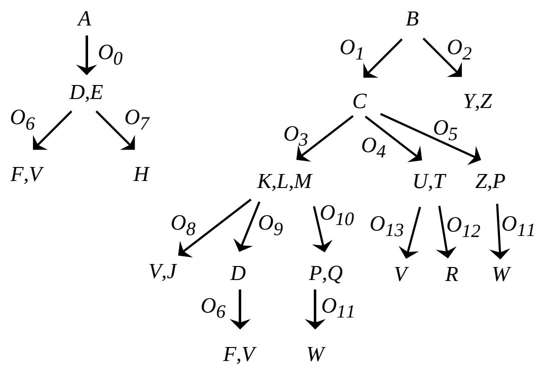

Figure 2: Task Refinability Graph

tons to ensure sketch compliance for generated plans.

Plan Skeletons

The abductive chains for a given anchor A can be used to define a set of labeled directed acyclic graphs called the *goal-anchor graphs* for A, which encode the connections from A to $\mathcal{G}$. To this end, we first define the *prefix-merge* of a set of chains.

Define a *prefix* of length n for a chain to be a path of length n from the root. We say two chains have a *common prefix* of length n iff the nodes and edges on the prefix of length n for those chains have the same labels. We say that a common prefix for a set of chains is *maximal* iff there is no common prefix of greater length. It is straightforward to show that for any set of chains having a longest common prefix of length k, either there are no edges emanating from the terminal node of the prefix on any chain, or all such edges are distinct. Define the *prefix-merge* of a set of chains to be the directed graph containing all the nodes and edges in the chains, with the following modifications:

- every maximal common prefix P is mapped to a single prefix P' whose nodes and edges match P
- the outward edges of the terminus of P' consist of the collection of outward edges from the terminus of each instance of prefix P.

Definition 4 (Goal-Anchor Graph Set) *The* goal-anchor graph set *for an anchor* A *is the collection of directed acyclic graphs that result from the prefix-merge of the inverted abductive chains for* A.

Because the goal-anchor graph set is built on top of the inverted abductive chains for an anchor, it encodes all possible refinements of goals in $\mathcal{G}$ that include the anchor. We denote the goal-anchor graph set for an anchor A by $GA\text{-}Set(A)$. The graph in $GA\text{-}Set(A)$ rooted at goal G is denoted by $GA(A;G)$.

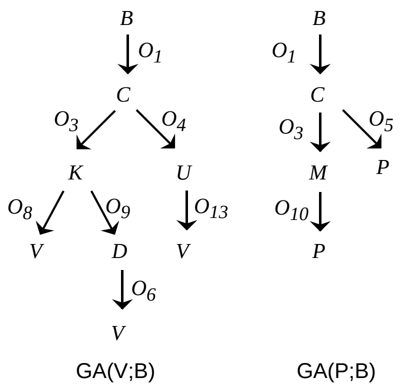

Figure 3: Sample Goal-Anchor Graphs

Example 2 Figure 3 presents the goal-anchor graphs for goal B in $GA\text{-}Set(V)$ and $GA\text{-}Set(P)$.

To this point, we have considered connections from individual anchors to $\mathcal{G}$ in isolation. To refine a given candidate goal, it is necessary to consider constraints imposed by any anchor that abstracts to that goal. We define the *plan skeleton* for a goal G to be the aggregate of all non-empty goal-anchor graphs for G in the goal-anchor graph sets of all anchors in the plan sketch. Plan skeletons encode the set of choices (for operators and variable instantiations) that are compatible with the user's plan sketch. For this reason, we use plan skeletons to inform the selection of operators and variable bindings during task refinement.

Task Refinement

For a given plan sketch, refinement of a candidate intended goal set $\{G_1, \ldots G_n\}$ begins with the creation of the plan skeleton Sk_i for each G_i. Planning then proceeds in traditional top-down fashion, with two modifications: *sketch enforcement* and *propagation of skeletons.*

Sketch enforcement relies on the selection of an appropriate *slice* of the skeleton for a given goal. A slice corresponds to a set of edges, one from the root of each goal-anchor graph in the skeleton for the goal, with the same operator and compatible bindings.

Definition 5 (Slice) *Let* Sk $= \{GA_1, \ldots, GA_n\}$ *be a plan skeleton for a goal G. A* slice *of* Sk, *denoted by* sl, *is a list of edges* $\{E_1, \ldots, E_n\}$, *such that for some operator schema O:*

- *the source node for each* E_i *is the root of* GA_i
- *the label of each* E_i *has the form* $\langle O, \sigma_i \rangle$
- *the* slice binding $\beta_{\text{sl}} = \bigcup_{1 \le i \le n} \sigma_i$ *is well-defined.*

The common operator schema for the slice edges is referred to as the *slice operator schema*. The slice operator schema instantiated by the slice binding is called the *slice operator*.

A slice represents a particular approach to refining a task that is consistent across all goal-anchor graphs of a skeleton. Application of the slice operator ensures that any solution produced (if there is one) will be in compliance with the original plan sketch. In general, there may be multiple slices for a given plan skeleton. Nonexistence of a slice indicates that the current task cannot be refined in accord with the original plan sketch (given previous plan refinement choices).

Relevant skeleton information must be propagated to subtasks that are created during the refinement process. Selection of a given slice commits to a certain approach; thus, all other approaches can be discarded. Furthermore, each edge in the slice maps to an individual task. Thus, a given task network node n need inherit only from those GAs whose associated slice-edge terminates in a node whose task matches $\mathit{Task}(n)$.

Definition 6 (Subtask Plan Skeleton) *Let n be a node in the task network generated by refinement of a goal G with plan skeleton $\mathrm{Sk} = \{GA_1, \ldots, GA_n\}$, using slice $\mathrm{sl} = \{E_1, \ldots, E_n\}$.*

- *The* successor *for $\langle G, \mathrm{Sk} \rangle$, denoted by* $\mathrm{SuccSkeleton}(\langle G, \mathrm{Sk}\rangle, \mathrm{sl})$*, is $\{GA'_1, \ldots, GA'_n\}$ where GA'_i is the subgraph of GA_i rooted at the terminus of E_i.*
- *The* plan subskeleton *for n, denoted by* $\mathrm{SubSkeleton}(n, \mathrm{Sk}')$*, is the set of GA'_i in* $\mathrm{Sk}' = \mathrm{SuccSkeleton}(\langle G, \mathrm{Sk}\rangle, \mathrm{sl})$ *whose root matches* $\mathrm{Task}(n)$.

Example 3 Consider refining the task B using the plan skeleton $\mathit{Sk} = \{GA(V;B), GA(P;B)\}$ (see Figure 3). The only possible slice for Sk has the operator O_1, which reduces B to C. The successor skeleton Sk_C consists of the subgraphs of $GA(V;B)$ and $GA(P;B)$ rooted at C. For Sk_C, the only possible slice consists of the edges from C labeled by O_3. The terminal nodes for these edges differ (K *vs.* M), reflecting the fact that O_3 is used for different purposes in the two goal-anchor graphs. Sk_C reduces C to the tasks $\{K, L, M\}$. At this stage, the plan skeleton is split: skeleton Sk_K for task K consists of the subgraph of $GA(V;B)$ rooted at K, while skeleton Sk_M for task M consists of the subgraph of $GA(P;B)$ rooted at M. Task L inherits no skeleton. Note that skeleton Sk_K has multiple possible slices, one consisting of the edge labeled O_8, and a second consisting of the edge labeled O_9.

There are two possible refinements of B given Sk, and hence two completions for the sketch $\{P, V\}$ using the intended goal set $\{B\}$. One contains the primitive tasks $\{V, J, L, W, Q\}$, the other $\{F, V, L, W, Q\}$.

Sketch Completion Algorithm

Figure 4 outlines our algorithm for the abductive completion of a plan sketch, given a set of operator schemas $\mathcal{O}$ and an initial world state S_0. Steps 1–3 perform abductive goal recognition, steps 4–5 create the initial task network for the chosen goals along with their associated skeletons, and step 6 performs sketch-compliant refinement of the created task network. Steps 1–5 were described earlier. Here, we focus on step 6.

Standard HTN planning can be characterized as a recursive algorithm on task networks. When the network contains only primitive tasks, the algorithm returns a *realization* of the task network if one is defined (*i.e.*, resolve all conflicts and instantiate variables), else the algorithm fails. When the network contains nonprimitive tasks, one is expanded by selecting an operator schema that matches an unsolved task and whose preconditions are satisfiable, applying it to generate an expanded task network, and then recursively invoking the algorithm on the new network.

Figure 5 outlines our sketch-compliant task refinement algorithm. It adheres closely to standard HTN refinement algorithms, making use of several standard HTN planning capabilities. $\mathit{Realize}(\tau)$ produces a completion of the task network τ if one exists. $\mathit{SelectOpr}(\mathcal{O}, n, \tau)$ nondeterministically chooses an instantiation of an operator schema from the set $\mathcal{O}$ that can be applied to the $\mathit{Task}(n)$ in task network τ. This function first identifies the schemas in $\mathcal{O}$ whose purpose matches $\mathit{Task}(n)$, using the function $\mathit{Match}(g_1, g_2)$. Any one of those schemas, O, whose preconditions are satisfied for the relevant bindings, as computed by $\mathit{Satisfied}(\mathit{Preconds}(O)^\sigma, n, \tau)$, can be returned. The function $\mathit{Reduce}(n, O^\sigma, \tau)$ refines a task network by expanding a node n by the (possibly partially) instantiated subtasks of O, adding ordering and world constraints as appropriate.

Our algorithm differs from standard HTN processing primarily in the operator selection process. *SelectOprFromSlice* returns an operator from a nondeterministically chosen slice of the plan skeleton of the task being refined, along with the successor skeleton Sk' for that slice. Sk' is passed as an additional argument to the function *Reduce* to propagate appropriate subskeletons to the newly created subtasks.

It is straightforward to establish the following result.

Proposition 1 $\mathrm{CompleteSketch}(\mathrm{Sk}, \mathcal{O}, S_0)$ *is a plan-sketch compliant algorithm that will nondeterministically find a primitive task network if one exists.*

```
$CompleteSketch(Sketch, \mathcal{O}, S_0)$

1. $Chains \leftarrow GetAnchorChains(Sketch)$

2. $Candidate\text{-}Goals \leftarrow Leaves(Chains)$

3. Nondeterministically select $I = \{G_1, \ldots G_n\}$ from
   $IntendedGoals(Candidate\text{-}Goals)$

4. Create plan skeletons $Sk_1, \ldots, Sk_n$ for $I$

5. $\tau_0 \leftarrow \langle \{(G_1, Sk_1), \ldots, (G_n, Sk_n)\}\{\}, \{\}\rangle$

6. Return $SolveSk(\langle \mathcal{O}, S_0, \tau_0\rangle, \tau_0)$
```

Figure 4: Abductive Completion of Plan Sketches

Analysis

Define the *abstraction factor* for task T to be the number of operator schema subtasks that unify with T. Also, define α be the maximum abstraction factor for all tasks in a domain. Let l_A be the difference in abstraction level between anchor A and the most abstract goal in the goal space $\mathcal{G}$. The sum $\Sigma_{A \in Sketch} \alpha^{l_A}$ is a loose upper-bound on the construction time for the set of abductive chains $\mathcal{C}$. Plan skeletons can be generated from the chains in time $l_A \times \mid \mathcal{C} \mid$.

While one should be wary of exponential bounds, the time required in practice should be small. In particular, one would expect anchors in plan sketches to be at intermediate levels of abstraction, and even the most complex planning domains rarely contain more than 8-10 abstraction levels. In our experiences with a range of domains (crisis recovery, mobile robots, air campaigns), abstraction factors for mid- to high-level tasks rarely exceed 6, and are often much lower.

One might expect that the use of plan skeletons to guide refinement would increase the time required for planning. However, plan skeletons can *reduce* overall task refinement time in certain cases, through early commitment to variable bindings, and the elimination of irrelevant operators. In general, the impact depends on the structure and solution density of the underlying search space. Still, the overall sketch completion time may be significantly higher than for pure HTN planning because of the possible need to consider multiple intended goal sets before finding one that can be refined to a sketch-compliant plan.

Prototype

We have implemented our abductive plan sketch completion method in a system built on top of the SIPE-2 generative planner (Wilkins 1988). The system accepts a user sketch and generates a fully-realized plan that

```
$SolveSk(\mathcal{P} = \langle \mathcal{O}, S_0, \tau_0\rangle,\ \tau = \langle T, L, W\rangle)$
  If $\tau$ is primitive
    Return $Realize(\tau)$ if defined,
    Else FAIL
  Else for a nondeterministically selected
    nonprimitive node $n \in T$
      $\langle O^{\sigma}, Sk'\rangle \leftarrow SelectOprFromSlice(\mathcal{O}, n, \tau)$
      $\tau' \leftarrow SolveSk(\mathcal{P}, Reduce(n, O^{\sigma}, \tau, Sk'))$
      Return $\tau'$

$SelectOprFromSlice(\mathcal{O}, n, \tau)$
  Return $\langle O^{\sigma'}, Sk'\rangle$ for nondeterministically selected
    $\langle O, \sigma\rangle \in Slices(Skeleton(n))$ where:
      $\sigma'' \leftarrow Match(Task(n), Purpose(O)^{\sigma})$
      $\sigma' \leftarrow Satisfied(Preconds(O)^{\sigma''}, n, \tau)$
      $\sigma' \neq \perp$
      $Sk' \leftarrow SuccSkeleton(\langle Task(n), Sk\rangle, \langle O, \sigma\rangle)$
```

Figure 5: Skeleton-based Plan Refinement

incorporates the sketch recommendations. Sketches can be specified as a list of tasks, or created interactively using a graphical plan editing tool.

The system has been tested in a legacy application for crisis action planning (Wilkins & Desimone 1994). The domain is reasonably complex, containing 45 operator schemas spread over 7 abstraction levels. The average task abstraction factor is 3.25. Final plans range in size from 15 to 250 nodes, distributed over numerous parallel threads of activity.

Experimental evaluation of planning algorithms within a single domain is not generally meaningful because the results are highly conditioned on the structure of the search space and the specifics of the goals to be solved. With that caveat, we present results for the crisis action domain that demonstrate the computational viability of the abductive sketch completion method in real domains. In order to evaluate the cost of the completion process, we tested the implementation using sketches designed to yield the same plan that the underlying planner generates when no sketch is provided. The sketches contained 6 tasks in the mid-range of the abstraction hierarchy (3-4 abstraction levels down from the toplevel tasks). Tasks at those levels are the most natural for defining plan sketches in this domain. The resultant plan contained 245 nodes. Tests showed that the cost of the completion process itself was negligible (roughly 1% of the overall plan generation time). We note that the evaluation solved a specific high-level goal set, thus excluding costs associated with checking multiple candidate intended goal sets for a sketch (which will generally be required).

Related Work

The TRAINS system (Ferguson, Allen, & Miller 1996) enables users to interactively guide the construction and execution of a plan through a cooperative, mixed-initiative effort. However, users are limited to inserting primitive tasks into a plan, and models of intended user goals are not created to guide plan refinement.

The DITOPS system (Smith & Lassila 1994) provides a kind of *schedule sketching* capability that shares many of the underlying motivations of the work presented here. In DITOPS, users can specify certain components of the overall schedule, with the system filling in the remainder in a consistent manner.

Our plan recognition model departs from previous work (*e.g.*, (Kautz 1990; Gertner & Webber 1996; Lesh & Etzioni 1996)) in two important ways. First, recognition proceeds from nonground tasks that span multiple abstraction levels. Second, recognition is tailored toward the production of an overall plan that is compliant with the events that underlie plan recognition. In particular, our approach requires that a candidate set of intended goals for a sketch be *validated* through generation of a completed plan that incorporates the elements of the sketch.

Conclusions

This paper has presented a framework for the abductive completion of user-supplied plan sketches. The framework employs a two-phase approach: an *interpretation* phase first identifies candidate sets of intended goals using abductive plan recognition techniques, then a *plan refinement* phase expands individual goal sets until it produces a full plan that is compliant with the original sketch. To the best of our knowledge, this work is the first plan sketch environment that supports completion of partial, multiple abstraction level sketches by reasoning about intended goals.

The work described here is part of a broader effort to make AI planning technology more accessible and easier to use through the metaphor of advice-taking (McCarthy 1958; Myers 1996a). Plan sketches constitute one form of user advice. Previously, we developed a theory of *strategic advice*, which amounts to user recommendations on how to refine individual tasks in an overall planning problem (Myers 1996b). Together, plan sketches and strategic advice enable substantial user-customization of the solutions generated by an automated planning system. Both capabilities are embodied in the *Advisable Planner* system (Myers 1996c).

Acknowledgments This research has been supported by DARPA Contract F30602-95-C-0259 as part of the ARPA/Rome Laboratory Planning Initiative.

References

Erol, K.; Hendler, J.; and Nau, D. S. 1994. Semantics for hierarchical task-network planning. Technical Report CS-TR-3239, Computer Science Department, University of Maryland.

Ferguson, G.; Allen, J.; and Miller, B. 1996. TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third International Conference on AI Planning Systems.* AAAI Press.

Gertner, A. S., and Webber, B. L. 1996. A bias towards relevance: Recognizing plans where goal minimization fails. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence.*

Kambhampati, S., and Hendler, J. 1992. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence* 55(2):192–258.

Kautz, H. 1990. A circumscriptive theory of plan recognition. In *Intentions in Communication.* Cambridge, MA: MIT Press.

Lesh, N., and Etzioni, O. 1996. Scaling up goal recognition. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96).* Morgan Kaufmann Publishers.

McCarthy, J. 1958. Programs with common sense. In *Mechanisation of Thought Processes, Proceedings of Symposium of the National Physics Laboratory,* 77–84. London, U.K.: Her Majesty's Stationery Office.

Myers, K. L. 1996a. Advisable planning systems. In Tate, A., ed., *Advanced Planning Technology.* Menlo Park, CA: AAAI Press.

Myers, K. L. 1996b. Strategic advice for hierarchical planners. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96).* Morgan Kaufmann Publishers.

Myers, K. L. 1996c. *User Guide for the Advisable Planner.* SRI International, Menlo Park, CA.

Smith, S. F., and Lassila, O. 1994. Toward the development of flexible mixed-initiative scheduling tools. In *ARPA/Rome Laboratory Planning and Scheduling Initiative Workshop Proceedings.* Morgan Kaufmann.

Thaler, D. E., and Shlapak, D. A. 1995. Perspectives on theater air campaign planning. Technical report, Rand Corporation.

Wilkins, D. E., and Desimone, R. V. 1994. Applying an AI planner to military operations planning. In *Intelligent Scheduling.* Morgan Kaufmann.

Wilkins, D. E. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm.* Morgan Kaufmann.