A Planner Independent Approach to Human Interactive Planning

Hyeok-Soo Kim kimhs@ict.usc.edu
Jonathan Gratch gratch@ict.usc.edu

University of Southern California and
Institute for Creative Technologies
Marina del Rey, CA, USA

Content Areas: AI planning, mixed-initiative system, human computer interaction, user interfaces

Abstract

This article introduces a novel approach to enhance the generality and modularity of human-interactive planning systems. We discuss the notion of planner-independent systems, whereby the planner underlying a human-interactive system is treated as a modular component that can be easily replaced depending on the characteristics of the application or as improved techniques become available. This is realized in a computation system that maps complex human planning requests into a series of primitive planning problems that can be answered by generic non-interactive planning system.

1 Introduction

This paper presents techniques to enhance the generality and modularity of human interactive (e.g., mixed-initiative) planning systems. By allowing human users and planning systems to jointly develop plans, such systems increase user's trust in the end product as well as complement the intuitive planning skills of human experts with the superior bookkeeping capabilities of automated planning systems. Indeed, human interactive planning systems have been applied to a number of significant applications including disaster planning [Allen et al., 2001], and tutoring systems for teaching people how to plan and make decisions [Rickel et al., 2002]. These systems differ in terms of who has authority or initiative, what knowledge is available to the agent, and the high-level communicative goals of the system (e.g., tutoring goals vs. task goals) [Hearst, 1999]. But they share a great deal, in particular the need to plan, including the ability to generate plans, provide feedbacks on the feasibility of different options, monitor their execution, and replan in response to unexpected events or user interventions.

These are large and complex systems that tightly integrate a number of capabilities. Beyond plan generation, they communicate with the user, often through natural language, model aspects of the user's mental state, recognize user intentions, execute plans and monitor the external environment. Further, users often demand flexible control over the planning process, at times micromanaging and at other times expecting the system to handle all the details. Facilitating this close and varying level of control over the planning process complicates the problem of cleanly separating the communication module from the planning process. For example, rather than simply accepting a goal and initial state, the planner must support a wide range of possible inputs. This lack of modularity limits the generality of these methods and contributes to the obsolescence of their component technologies. This is most obvious with regard to the planning techniques that underlie these systems, arguably their most essential component technology.

Due to this tight connection between planning and communication components, most planning systems underlying human interactive planners are antiquated or rudimentary. Planning techniques are, on the other hand, evolving rapidly. A quick review of the recent AI Planning systems competitions illustrates that the top performing planners are in constant flux. For example, in 1998, IPP was the winner of the ADL track and also showed good performance in the STRIPS track, and HSP solved the most problems in the STRIPS track [McDermott, 1998]. In 2000, IPP was replaced by FF (faster but total order) [Bacchus, 2000]. In 2002, MIPS solved the highest number of problems in the fully automated track, and was the only system that produced solutions in each track, while FF also out-performed its competitors in the numeric and STRIPS domains [IPC, 2002]. To our knowledge, none of these techniques have been incorporated into state-of-the-art human interactive planning systems.

Tying a human interactive planner to a specific planning algorithm can also significantly limit its generality. The developers of planning systems point to the *domain-independence* of their planning techniques as an argument for their wide applicability. There is a strong reason to doubt this claim. Planning performance varies dramatically across application domains. Different planning systems excel on different domains and some, supposedly domain-independent, planners cannot even represent distinctions essential for certain domains. For example, at the AI plan-

ning system competition in 2002, FF planner exhibited outstanding performance against its competitors in most of the Numeric and STRIPS problems, but it didn't compete in the temporal domains. On the other hand, TALPlanner outperformed its competitors in the temporal domains, but didn't participate in the numeric domains [IPC, 2002].

We argue that *planner-independence* is a far more crucial design goal than domain-independence in the design of human interactive planning systems, though is not immediately obvious how to achieve this goal. A planner-independent system embodies the design philosophy that the planner should be a modular component that can be easily replaced depending on the characteristics of the application or as improved techniques become available. Unfortunately, due to close input a user has over the planning process, planning and user-interface modules are tightly intertwined in human interactive planning systems. In contrast, the typical interface to conventional planning system is at the level of specifying a domain theory, initial state and a set of goals, which is really an inappropriate level to separate planning and communicative functions.

The question we address in this paper is what is the right level of abstraction (what is the right API) to separate the planning system from other modules in a human interactive planning system, specifically focusing on the connection to the user-interaction module, which tends to have the tightest interaction. Our approach is to consider the generic planning services necessary to support collaboration, define a level of abstraction in terms of these basic services, and map between these services and the low-level API of planning systems in a planner-independent fashion.

Section 2 lays out a high-level description of human interactive planning systems and section 3 elaborates our basic approach. Section 4 describes how we map from the basic planning services necessary for collaboration to low-level calls to a conventional planning system.

2 Human Interactive Planning Systems

Human interactive planning systems allow a human user and an intelligent system to closely interact during the process of plan generation, execution, and repair. Such systems have been explored in the research community under a variety of titles including mixed-initiative planning systems, human interactive planning systems, and tutoring systems. They differ in many respects but they share two tightly intertwined capabilities: they must be able to develop and reason about plans, and they must communicate with the user about the planning processes. Here we review this work in terms of terminology we will use in this article.

Communication

The interaction between a user and a system can be seen as a dialogue, and researchers in this area have been heavily influenced by linguistic theories, regardless of whether the system actually communicates via natural language or through a more stylized interface. In such systems, the interaction is controlled by a *dialogue manager* and the content of the interaction is frequently characterized in terms of *speech acts* [Austin, 1962], a formulism for characterizing and classifying natural language utterances. For example, request like “where is the helicopter” is represented as an “information request” about some attribute of a domain entity. Speech acts have a certain structure and impose certain communicative obligations on the listener. For example, upon receiving an information request, the system is obligated to respond to the request with some assertion. Standard speech acts include *inform, order, request, accept, reject,* and *counter-propose*. Depending on the system’s capabilities, these are frequently augmented with *dialogue acts* that manage turn-taking and shifts in initiative between the system and the user [Traum, 1999].

Speech acts provide an abstract structure, but by analyzing human-to-human collaborative planning dialogues, collaborative planning researchers have also classified general features of the content of these conversations. Human interactive planning dialogues revolve around aspects of planning process, including the development and refinement of courses of action, the evaluation and comparison of alternatives, the clarification of features of the environment, the identification of plan problems or threats, and the clarification of aspects of the planning dialogues. By combining these aspects with speech acts, these systems can classify a range of utterances. For example, a user might inform the system of a course of actions, order the system to adopt it, request the system to develop an alternative, accept the alternative, order its execution, etc.

Planning and Planning Service Requests

Speech act theory facilitates understanding, but to service such speech acts, the system must also support a range of plan reasoning capabilities. If a user requests a course of actions, the system must be able to develop one. If the user requests a comparison of alternatives, the system must have the means to provide it. If the system wishes to take initiative this must be motivated by some inferences concerning the planning process.

We use the term *planning service requests* to refer to the collection of planning capabilities that are necessary to address a user’s plan-related speech acts and to inform the systems inferences about dialogue initiative. Collectively, planning service requests define an API that the underlying plan reasoning system must support. Unfortunately, from the perspective of utilizing traditional AI planning systems, the API required to support human interactive planning differs a great deal from the standard interface to one of these traditional systems. Rather, these have focused on turnkey solutions to the plan generation problem. The planner is treated as a black box that accepts a “planning problem” – a domain theory, and initial and goal state descriptions – and returns a single satisfying plan. With the possible exception of search control heuristics or some intermediate state con-

straints, the actual planning process is opaque and not subject to outside manipulation.

In contrast, human interactive planners must support much richer planning process. In terms of plan generation, users frequently demand tighter and incremental control over the planning process. As in many real-world planning domains, plans are typically hierarchical and users interact with the system to refine their plans, explore or get advice about different courses of action, and receive assistance in initiating and monitoring the plan's execution. Consider the following hypothetical interchange from a military decision-making application we have been developing (Rickel et al., 2002):

USER: How can I reinforce the platoon in downtown Celic?
SYSTEM: There are two feasible options. Send two squads forward or send one squad to secure our route and speed our subsequent movement.
USER: What is the disadvantage of sending two squads?
SYSTEM: It will fracture our forces and limit our future options.
USER: Send first squad to secure the route.
SYSTEM: Sir. First squad needs to secure a landing zone. I suggest we send forth squad.
USER: We don't need to secure the landing zone anymore. Send first squad.
…

This example illustrates several planning service requests that do not obviously map to traditional planning problems. Planning is hierarchical and incremental rather than generating a complete plan, with the user or system proposing single step refinement to the plan. Rather than generating a single satisfying plan, multiple qualitatively different options are explored in parallel (e.g., sending one versus two squads). Rather than receiving a simple list of plan steps, the user may ask for evaluative information (e.g., what is the disadvantage), and the system may take the initiative to offer advice (e.g., first squad is preoccupied). Finally, rather than accepting a fixed goal state, planning requirements may change in mid-stream (e.g., as when the user drops the goal that the landing zone be secure). Human interactive planners typically also incorporate functions to execute plan steps, monitor their execution, and detect when plan repairs are necessary.

3 Planner-Independent Human Interactive Plan Assisting System (HIPAS)

Figure 1 illustrates our view of how to impose greater modularity between the planning and the communicative modules in a human interactive planning system. The key idea is to define a set of abstract planning service requests that can serve as a bridge between the dialogue manager and a traditional planning system. Under this view, the question we must address is how to provide a general and planner-independent mapping between these service requests and the capabilities of planning systems.

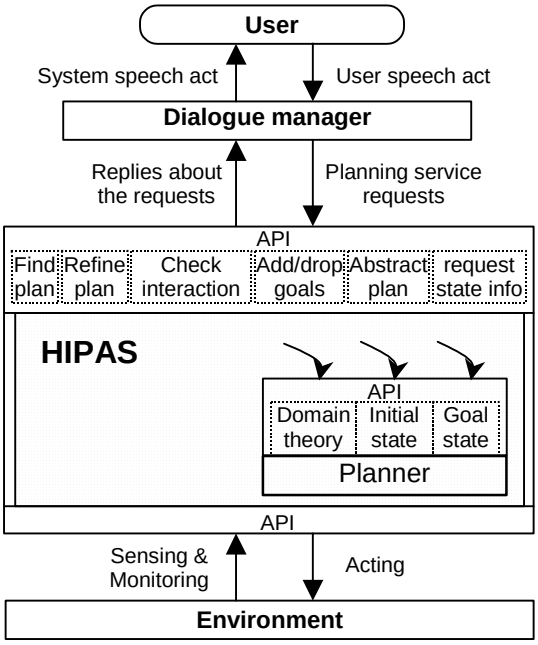

Figure 1: HIPAS architecture

Our approach is motivated by the "in practice" efficiency of recent planning techniques within the context of collaborative planning domains. Although planning in general is hard (indeed, undecidable), for suitably constrained applications recent planning techniques can, in practice, "solve" the planning problem. This is particularly true for the relatively constrained propositional domain theories used by many human-interactive systems. For example, in the MRE (Mission Rehearsal Exercise) team training system, a restricted plan scripting language sufficed to drive mixed-initiative interaction [Rickel et al., 2002]. We discovered that IPP, a contemporary planning system, could do full plan generation on the same domain theory in less time than it took to find a scripted solution (typically less than 100 ms to generate a plan from scratch). Further, our early experiments with JADE, one of the most complex human-interactive planning domains, shows that IPP can solve problems in at most a couple of seconds [Cox and Veloso, 1997]. This is not to say that planning is a solved problem, but for many of the domains considered by human interactive planning, these new planning techniques are far more than adequate, and rather, it is the human-to-agent communication that serves as the main temporal bottleneck.

Our admittedly strong assumption in this paper is that planning is decidable and relatively efficient in practice, though we discuss how to relax this assumption later in the paper. If plans can be generated in milliseconds, however, it opens up new possibilities for human interactive planning. One could imagine repeatedly calling a planner to solve variations of a planning problem to explore features of a

domain. For example, if the user wanted to consider the differences between evacuating injured personnel with a helicopter versus an ambulance, one could simply solve the planning problem twice – once with the domain theory without the helicopter action and once without the ambulance action– and summarize the outcomes to the user.

We build on this observation, showing how a more flexible repertoire of planning service requests can be constructed by solving a set of suitably varied planning problems. By systematically varying a traditional planner's domain theory, initial state and goal state, the system can, by brute force, provide planner-independent mappings between the planning service requests demanded by human interactive systems and the capabilities of traditional planners. The planning system itself can be treated as a black box and alternative planners could be incorporated, assuming they all take as input an initial and goal state description and a domain theory consisting of a set of primitive actions.

4 Mapping planning services into traditional planning problems

Here we describe how HIPAS maps high-level planning service requests (e.g., refine the current plan or check interaction) onto a sequence of traditional planning problems. To handle such requests, the system needs a mechanism that maps them into appropriate inputs for a conventional planning system and also provides the dialogue manager interpretable feedback from the result of the planning system. First we introduce new representational constructs, *hierarchical action sets* and *the current plan set*, to facilitate this mapping.

4.1 Hierarchical action set

A hierarchical action set is a novel domain representation that takes advantage of the speed of modern non-hierarchical planning algorithms but retains the control and flexibility of human-interactive hierarchical planning. It is a tree-like AND/OR graph that consists of both abstract and primitive actions and it represents hierarchical decomposition rules that refine a high-level action to a set or multiple alternative sets of lower-level ones.

Though superficially similar to hierarchical action structures in conventional hierarchical planning systems [Erol et al., 1994], there are significant differences. An abstract action represents an unordered *set* of primitive actions that are potentially useful for achieving a goal rather than a high level *sequence* of actions to perform. One can only recover a valid subplan from an abstract action by feeding the set of primitive actions to a traditional planner along with a goal and initial state. Decomposing an abstract action corresponds to building the set of primitive actions into several more focused subsets, rather than yielding a more detailed lower level description.

For example, consider the hierarchical action set for obtaining a shelter shown in Figure 2. Two different decompositions of the root action, rent an apartment and buy a house, subdivide the entire set of six primitive actions to two smaller and qualitatively distinct subsets, {searching classified ads, visiting apartments, placing deposit} and {getting a real estate agent, getting loan pre-approval, visiting open houses}, respectively. If one prefers to rent an apartment, the system retains the primitive actions under rent an apartment in the current domain and excludes the primitive actions related to buying a house. At any point in the plan refinement process, the current set of primitive actions is passed to a non-hierarchical planning system to check the existence of a complete plan. This process allows a user to hierarchically construct a plan under his/her control and preference while continually reorganizing the domain theory to reflect to his/her choices.

For a given domain, there can be multiple hierarchical action sets, each of which has its own goals. These goals are used for selecting appropriate hierarchical action sets according to the user's goals. For example, let's assume there are four actions sets, A, B, C, and D, and each has a goal, a, b, c, and d respectively. When a user wants to generate a plan to achieve b and d, the system will automatically select B and D as initial action sets. To avoid redundancy and ambiguity in selecting initial action sets, a set of goals associated with an action set cannot be a subset of goals of another action set, but two action sets can have goals in common. Hierarchical action sets satisfy the downward and the upward solution properties, so once an abstract solution is found all other abstract plans can be pruned away and all the descendents of any inconsistent abstract plan can be pruned away as well [Russell and Norvig, 1995].

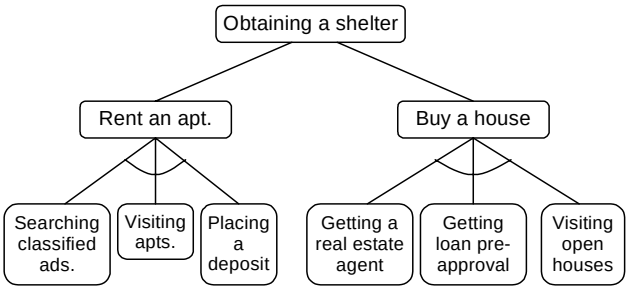

Figure 2: An example of a hierarchical action set for obtaining a shelter

4.2 Current plan set

To keep track of development of a plan while exploring hierarchical action sets and frequently changing the current domain theory, the system uses an array, *the current plan set*, to represent the current plan developed so far. Initially, it consists of the root actions of selected action sets, such that the union of the effects of the root actions should contain all of the user's goals. If different combinations of action sets could satisfy these goals, these are considered potential choice points for the user. The system first checks

whether or not each combination has a solution that is consistent and complete by sending the planner a new domain theory that consists of all the primitive actions in each combination in turn. If the planner finds a solution to the user's goals with a given combination of sets, this will be considered a valid alternative for solving the problem. The dialogue manager can then present these alternatives to the user, requesting him/her to choose one. After one among the survived candidates is selected by a user, its root actions become members of the initial current plan set.

The current plan set is updated by replacing each abstract action with one of its refinements selected by a user after checking for consistency between the newly introduced actions and the existing ones. This process can continue until all the actions in the current plan set are primitive. Whenever the current plan set is updated, the hierarchical action sets are also reorganized by pruning away unselected or inconsistent branches.

The above two new domain representations make it possible for users to select their own strategies rather than the system imposing a specific refinement strategy, such as least commitment or fewest alternative first (FAF) [Reiko et al., 1997]. The system just provides the necessary information to implement these strategies, i.e., the number of alternative refinements or the consistency of each refinement with the rest of the current plan.

4.3 Solution matrix

HIPAS handles planning service requests by systemically solving variants of the current plan set and combining the result in formulating an answer to the request. The solution matrix represents this computation.

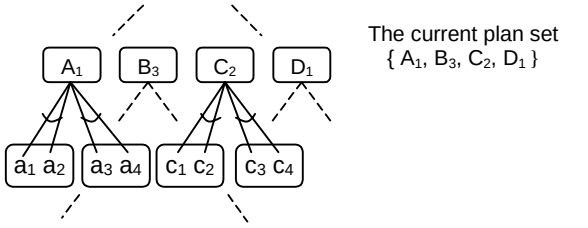

Figure 3: Decompositions of A1 and C2

A_1	B_3	C_2	D_1	consistency
(a_1, a_2)	All the primitive descendents of B_3	(c_1 c_2)	All the primitive descendents of D_1	Yes
(a_1, a_2)		(c_3, c_4)		Yes
(a_3, a_4)		(c_1 c_2)		Yes
(a_3, a_4)		(c_3, c_4)		No

Table 1: A solution matrix of the compatibilities of A_1 and C_2

For example, in Figure 3, the user wishes to decompose two abstract actions, A_1 and C_2, in parallel (a user can decompose them simultaneously). HIPAS must infer which combinations of refinements are consistent with each other. The solution matrix in Table 1 illustrates how this service request is translated into a series of planning problems that are specializations of the current plan set. To check the compatibilities, HIPAS constructs a different domain theory for each candidate consisting of primitive actions and the primitive descendents of the abstract actions in the current candidate plan set. After examining each domain theory with a conventional planning system, HIPAS generates a solution matrix that summarizes the results. If the user chooses one among possible candidates, the current plan set is replaced with the candidate. For example, if the user chooses (a_1, a_2) and (c_3, c_4) the current plan set is changed from {A_1, B_3, C_2, D_1} to {a_1, a_2, B_3, c_3, c_4, D_1}.

4.4 Planning service requests

In the rest of this section, we describe how the high-level planning services are mapped to traditional planning problems in terms of hierarchical action set, current plan set, and solution matrix. Up to now we've focused on decomposition. Now we'll generalize this discussion to a number of service requests.

Find plan: when the system gets a request from a user to generate a plan for a given problem, HIPAS first identifies hierarchical action sets relevant to stated user goals. In the example in Figure 4, if the given goals are to achieve Q and Z, then there exist two possible sets, {A, B} and {A, C}. Note that there may be multiple consistent combinations of action sets that are treated as alternatives for the user to select amongst.

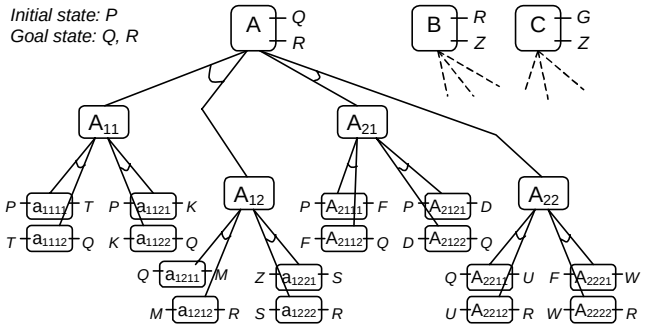

Figure 4: Action hierarchy. The preconditions appear left of the action, and the effects right

A_{11}	A_{12}	Consistency
All the primitive descendents of A_{11}	(a_{1211} a_{1212})	Yes
	(a_{1221}, a_{1222})	No

Table 2: Solution matrix for refining A_{12}

Refine plan: When the user requests to expand some abstract actions in the current plan set, HIPAS shows the direct descendents of these actions. If there are multiple choices for decomposing a selected action, a user can request to check the consistency of the different alternatives. From the example in Figure 4, let's assume that the current plan set is

$\{A_{11}, A_{12}\}$. If the user would like to expand A_{12} first, then the system checks the consistency of each alternative refinement of A_{12} with the remaining steps in the current plan (in this case, just the step A_{11}). HIPAS constructs primitive planning problems for each of these alternatives by specializing the domain theory in each case. For example, the first planning problem includes actions associated with a_{1211} and a_{1212} but not a_{1221} or a_{1222}. The solution matrix in Table 2 shows the consistency of the two different decompositions of A_{12} with the remaining plan steps. This table illustrates that (a_{1221}, a_{1222}) is inconsistent with $A_{11,}$ meaning that the user can pick only one decomposition, (a_{1211} a_{1212}). After the user chooses one of the consistent options, the current plan set is replaced to the newly selected one, i.e., from $\{A_{11}, A_{12}\}$ to $\{A_{11}, a_{1211}, a_{1212}\}$.

Check interaction: A key difficulties in human-interactive plan is detecting and communicating interdependencies between different portions of the plan. For example, how a user decides to obtain a shelter may constrain his or her options for obtaining transportation. A novel feature of HIPAS allows users to systematically explore interactions among alternative plan refinements by evaluating the consistency of combinations of possible refinements. For example, the solution matrix in Table 1 evaluates the pair-wise interactions between refinements of A_1 and $C_{2.}$ The results indicate an interaction between these tasks (how the user chooses to refine A_1 will limit his or her options for refining C_2). Although the system cannot detect exactly why the interaction occurred (the planner only returns that a combination failed, not why the failure occurred), it provide useful feedback to the user: e.g., "if you refine A_1 into a_3 and a_4, you will constrain your options for refining C_2". Depending on the time available, HIPAS can progressively deepen its interaction checks, initially exploring pair-wise interactions, moving on to 3-way interactions, etc.

Abstract steps: In addition to refinement, the system allows non-chronological backtracking over refinement decisions. When a user changes his/her previous decision, the related actions in the current plan set are replaced by their parents for reconsideration, as well as any siblings that were previously pruned away from the hierarchical action sets are recalled into the sets.

Add or drop goals: Users can add or drop goals in the middle of plan generation. When users want to drop goals, it doesn't invalidate the current plan set, though it may now contain unnecessary plan steps. These plan steps are automatically removed through the process of elimination of unnecessary plan steps. For example, let's assume that the original goals are [Q, R] and the current plan set is $\{A_{11}, A_{12}\}$. After dropping a goal, [R], A_{12} becomes irrelevant and is automatically removed from the current plan set.

On the other hand, if a user requests to add a goal, HIPAS finds an appropriate hierarchical action set that can achieve the goal, and then adds the root of the action set into the current plan set. In the example in Figure 4, if a user want to add a goal, [G], HIPAS add the action (C) to the current plan set. However, this may cause unnecessary processing when the goal can be achieved from the current plan without any additional plan steps. For example, let's assume the original goal is [Q] and the current plan set consists of $\{A_{11}, A_{12}\}$. If requested to add [R] to the current set of goals, HIPAS adds an action (B) to the current plan set. Since there are duplicated plan steps that achieve [R], one of them, A_{12} or B, should be removed from the current plan set. To prevent this problem, whenever the system gets a request to add goals, it first checks if the current plan can achieve the goals without any change. If not, then HIPAS finds and adds appropriate action sets to the current plan set.

There are two additional services that are applied regardless of user requests. The following two services are performed whenever a current plan set is updated and a plan step is executed respectively.

Eliminate unnecessary plan steps: As the user refines the plan, certain steps in the current plan can become unnecessary if their effects are fortuitously achieved by other parts of the plan. HIPAS automatically detects if there are any unnecessary plan steps. For example, in Figure 4, let's assume that the goal state is [Q] instead of [Q, R]. Since there is no hierarchical action set that achieve only [Q] the system should select {A} as an initial current plan set that is to achieve R as well as Q. Then, the current plan set is updated to $\{A_{11}, A_{12}\}$. Since A_{12} is just to achieve [R], it is unnecessary. HIPAS detects the existence of any plan step that has no primitive descendent in the returned plan from the planning system and marks this step as unnecessary. In the example, A_{12} is an unnecessary step, so it is removed from the current plan set, i.e., the current plan set is simplified from $\{A_{11}, A_{12}\}$ to $\{A_{11}\}$.

Monitoring and replan: HIPAS monitors the current world to detect external events. If the system detects any unexpected change in the current world, it examines the validity of the current plan with the change. When the plan is no longer consistent with the changed world state, the system must throw out the current plan and regenerate a new plan from scratch to achieve the user goals from the new situation. HIPAS cannot implement more sophisticated repair strategies as it cannot detect what caused the plan failure given it doesn't have any knowledge about internal planning process of the base planning system.

5 Conclusions

The current system is limited by the range of implemented planning service request and we are currently working to expand this repertoire. For example, users currently are unable to post ordering constraints between actions. Such requests could be handled by adding auxiliary constraints to the base planner's domain theory. For example, one way to enforce abstract action A to occur before abstract action B is to add "A-complete" effects to each of A's set of primitive actions and "A-complete" preconditions to each of B's ac-

tions. We believe that many additional requests can be covered by similar representational manipulations.

The current system depends on the strong assumption that the base planner can quickly either solve a planning problem or detect that no solution exists. While this may be reasonable for simple domains, it could be unreasonable in general. We can relax this requirement by extending solution matrices of HIPAS to a 3-value logic (consistent, inconsistent, unknown) where "unknown" means that the planner could not terminate within some small pre-specified time limit. We would then have to suitably qualify the results of planning service requests, making responses, in essence, heuristics rather than definitive results. For example, rather than stating that a refinement was inconsistent, HIPAS could state that it may be inconsistent. The user could increase the quality of this feedback by extending the time limit, or the system could periodically fill in solution matrices as time becomes available.

HIPAS is being applied within the context of the Mission Rehearsal Exercise human-interactive planning system, a rich virtual training environment that includes human trainees interacting with graphically embodied virtual agents that can communicate with through natural language about tactical decisions. HIPAS shows promise to extend the planning capabilities of the exiting MRE system while at the same time allowing the planning component to be more modular and easier to update with more advanced planning techniques as they become available.

Acknowledgements

This paper was developed with funds of the Department of the Army under contract number DAAD 19-99-D-0046. Any opinions, findings and conclusions or recommendations in this paper are those of the authors and do not necessarily reflect the views of the Department of the Army.

References

[Allen et al., 2001] J Allen, D Byron, M Dzikovska, G Ferguson, L Galescu, and A Stent. Towards conversational human-computer interaction. , *AI Magazine*, 22(4): 27--37, 2001.

[Austin, 1962] J.A. Austin. How to do things with words. Harvard University press, Cambridge, Massachusetts, 1962.

[Bacchus, 2000] Fahiem Bacchus. Results of AIPS-2000 planning competition. 2000. url:http://www.cs.toronto.edu/aips2000/SelfContainedAIPS-2000.ps.

[Cox and Veloso, 1997] Cox, M. T., and Veloso, M. M.. Controlling for unexpected goals when planning in a mixed-initiative setting. In E. Costa & A. Cardoso (Eds.), *Progress in Artificial Intelligence: Eighth Portuguese Conference on Artificial Intelligence*: 309—318, Berlin: Springer, 1997.

[Erol et al, 1994] Kutluhan Erol, James Hendler, and Dana S. Nau. UMCP: A sound and complete procedure for hierarchical task-network planning. In AIPS-94, pages 249--254, 1994.

[Hearst, 1999] Marti A. Hearst. Trends & Controversies: Mixed-initiative interaction. IEEE Intelligent System, 14(5):14--23, 1999.

[IPC, 2002] IPC, the results of 2002 International Planning Competition. 2002. url:http://www.dur.ac.uk/d.p.long/competition.html

[McDermott, 1998] D McDermott. The 1998 Planning system competition, Artificial Intelligence, 21(2):35--55, 2000.

[Reiko et al., 1997] Reiko Tsuneto, Dana Nau, and James Hendler. Plan-refinement strategies and search-space size. *Proceedings of the European Conference on Planning*, pages 414--426, 1997.

[Rickel et al., 2002] J Riclel, S Marsella, J Gratch, R Hill, D Traum, and W Swartout. Toward a new generation of virtual humans for interactive experiences. *IEEE Intelligent Systems* 17(4):32--38, 2002.

[Russell and Norvig, 1995] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. Prentice-Hall, Upper Saddle River, New Jersey, 1995.

[Traum, 1999] David R. Traum. *Speech acts for dialogue agents*. In Rao, A. and Wooldridge, M., editors, Foundations of Rational Agency. Kluwer, 1999.