Active Acquisition of User Models: Implications for Decision-Theoretic Dialog Planning and Plan Recognition

DEKAI WU*
Computer Science Division, University of California at Berkeley
Berkeley, CA 94720, U.S.A.
dekai@cs.berkeley.edu

(Received 16 October 1990; in final form 15 January 1991)

Abstract. This article investigates the implications of *active user model acquisition* upon plan recognition, domain planning, and dialog planning in dialog architectures. A dialog system performs active user model acquisition by querying the user during the course of the dialog. Existing systems employ passive strategies that rely on inferences drawn from passive observation of the dialog. Though passive acquisition generally reduces unnecessary dialog, in some cases the system can effectively shorten the overall dialog length by selectively initiating subdialogs for acquiring information about the user.

We propose a theory identifying conditions under which the dialog system should adopt *active acquisition goals*. Active acquisition imposes a set of *rationality requirements* not met by current dialog architectures. To ensure rational dialog decisions, we propose significant extensions to plan recognition, domain planning, and dialog planning models, incorporating decision-theoretic heuristics for expected utility. The most appropriate framework for active acquisition is a multi-attribute utility model wherein plans are compared along multiple dimensions of utility. We suggest a general architectural scheme, and present an example from a preliminary implementation.

Key words: active acquisition, decision-theoretic planning, decision theory, dialog planning, dialog systems, expected utility, multi-attribute utility, plan recognition, subdialogs, user modeling.

1. Introduction

In this article, we investigate the implications of *active user model acquisition* upon plan recognition, domain planning, and dialog planning in dialog architectures. Consider the following dialog between a user and a route consultant system:

---

* The author will be at the Department of Computer Science, University of Toronto, until Fall 1992, and at the Department of Computer Science, Hong Kong University of Science & Technology thereafter. This research was sponsored in part by the Defense Advanced Research Projects Agency (DoD), monitored by the Space and Naval Warfare Systems Command under N00039-88-C-0292, the Office of Naval Research under contract N00014-89-J-3205, and the Sloan Foundation under grant 86-10-3.

User Modeling and User-Adapted Interaction **1:** 149–172, 1991.

© 1991 *Kluwer Academic Publishers. Printed in the Netherlands.*

(1) U: How do I get to the center of the bay?
(2) S: Why do you want to go there?
(3) U: I want to take a picture of the skyline.
(4) S: Is it sufficient to drive to Treasure Island, or is it necessary to take a cruise?
(5) U: No, I don't want to take the picture from Treasure Island.
(6) S: Then you can take a bay cruise tour from Fisherman's Wharf.

This kind of dialog behavior, an example of active user model acquisition, is beyond most existing dialog systems. Not only does the user query the system, but the system also queries the user, in order to acquire information about the user that can be used to increase the effectiveness of its responses.

Active acquisition is a logical extension of research on dialog-based consultation [e.g., Wilensky et al. 1988, Wahlster et al. 1983, Kobsa 1985]. Previous user modeling systems have been either entirely passive or entirely active. GRUNDY [Rich 1979] relied exclusively on an active acquisition phase prior to the main consultation session, asking the user a "canned" set of questions. The "canned" nature of the acquired information restricted consultation applications to domains with little conceptual variation from one dialog to another. Later systems such as KNOME [Chin 1988], TRACK [Carberry 1988], and PAGAN [Mayfield 1989] were designed to extract user models by making inferences based on passive observation of the dialog. Passive acquisition is generally preferable to active querying, to minimize unnecessary dialog. However, we are concerned with cases where the system *should* actively initiate subdialogs for the purpose of acquiring information about the user. In our view, neither entirely active nor entirely passive strategies are adequate when reasonably flexible queries are permitted. Our theory was developed to identify conditions under which interactive acquisition is called for.

It has been possible in existing dialog architectures to use simple plan recognition and planning models, because the task of conducting dialog is simplified by performing user model acquisition entirely passively or entirely actively. These models, however, are insufficiently flexible to handle the sort of trade-off decision-making necessary for interactive acquisition. We argue in this article that major extensions to plan recognition, dialog planning, and domain planning models are needed, incorporating multi-attribute decision-theoretic notions of expected utility.

Active acquisition is strongly connected with plan recognition research [Allen and Perrault 1980, Cohen and Perrault 1979, Litman and Allen 1984] but has a different emphasis. Whereas plan recognition mechanisms only *recognize* the correlations between a given discourse structure and the conversants' plans, active acquisition models *predict* the most cooperative continuation of a dialog, given the conversants' plans.

In Section 2, the motivations for active acquisition are discussed, and an overview of the theory is given. We introduce the notion of *active acquisition goals* (AAGs) which cause the system to query the user, and consider a number of rules for generating AAGs in response to reasoning failures that occur in the normal operation of a dialog system. In Section 3, the decision-theoretic implications for the architecture of dialog agents are considered. We first note that the active acquisition theory requires the dialog architecture to satisfy a number of rationality conditions that are often not met by existing architectures. We then examine five dimensions of expected utility that play a role in generating and selecting active acquisition goals. Section 4 describes an architecture we are constructing as a preliminary implementation to meet these requirements. An example is followed in greater detail in Section 5.

2. Active Acquisition

An ideal dialog agent relies primarily on passive acquisition to construct a user model, to meet considerations of efficiency, cooperativeness, and appropriateness. For example, no active acquisition is required in the following dialog:

(1) U: Is there an inexpensive motel close to Fisherman's Wharf?
(2) S: Yes, but they are usually full at this time of the year. You might try the Ponderosa Inn, which is about a mile from Fisherman's Wharf. There is a cable car stop less than two blocks away.
(3) U: How do I get there?
(4) S: Go down the street, then turn right on Powell, at the third light. There is a sign on the left-hand-side after half a block.

To generate these responses, the system must build a user model containing many inferences, among them:

- The user has the goal of knowing an inexpensive motel close to Fisherman's Wharf.
- The user assumes the system has this knowledge, and wants the system to communicate it to him.
- The user has the goal of staying in the motel today.
- The user wants to be as close as possible to Fisherman's Wharf.
- The user wants to be able to reach Fisherman's Wharf as conveniently as possible.
- The user's budget constraints outweigh his desire to stay close to Fisherman's Wharf.

- The user has the goal of knowing how to reach the motel.
- The user is driving a car (because he asked for a motel).
- The user does not live in the area, and is probably a tourist.
- The user does not know the area well, and requires detailed directions.

By acquiring the user model through passive observation of the dialog, the system is able to generate responses that take all these factors into account without burdening the user with unnecessary querying. The same considerations of efficiency, cooperativeness, and appropriateness that dictate heavy reliance on passive acquisition, however, also dictate occasional *motivated* use of active acquisition. Consider again the earlier example:

(1) U: How do I get to the center of the bay?
(2) S: Why do you want to go there?
(3) U: I want to take a picture of the skyline.
(4) S: Is it sufficient to drive to Treasure Island, or is it necessary to take a cruise?
(5) U: No, I don't want to take the picture from Treasure Island.
(6) S: Then you can take a bay cruise tour from Fisherman's Wharf.

A passive system would respond:

(1) U: How do I get to the center of the bay?
(2a) S: You can drive to Treasure Island.
(3a) U: But I don't want to be on land.
(4a) S: Then you can charter a yacht.
(5a) U: But I don't have the money.
(6a) S: Then you can rent a sailboard.
(7a) U: But I can't windsurf.

and so on. Alternatively, the system could suggest all known options:

(2b) S: You can drive to Treasure Island, charter a yacht, rent a sailboard, take BART through the tube, swim in a wetsuit, take a bay cruise tour, buy a rowboat, scuba dive, hop on the ferry, or hire a helicopter.

Clearly, for some queries an active acquisition approach is more efficient, cooperative, and appropriate. The information needed to select the best plan—i.e., the knowledge that the user has a goal of taking a picture of the skyline from an angle other than Treasure Island—cannot be inferred passively. The need to know the user's specific goal becomes apparent only after the dialog has already begun, requiring post-query acquisition rather than GRUNDY's pre-session "canned" active acquisition. Sleeman's [1985] UMFE system actively acquires information about the user's expertise level after the user's query, but is limited to asking a series of yes-no questions. More flexible discourse structures are needed; the system's query (4) depends on the user's response (3) to the system's first query (2). Moreover, these active acquisition queries are useful for purposes other than determining the user's expertise level.

2.1. ACTIVE ACQUISITION GOALS (AAGS)

As a conceptually useful notion for studying subdialog initiation behavior, we define an *active acquisition goal* (AAG) as a goal held by a dialog agent to actively acquire knowledge about the dialog partner. Continuing with the previous example, the system must make the decision to initiate the clarification subdialog (2)–(3) so as to determine more specifically what the user's goal is. The system must also decide to initiate the subdialog (4)–(5) in order to determine the user's preference between two plans. We model this behavior by assuming that in normal operation, a dialog agent builds a user model using passive acquisition, but occasionally adopts an AAG, and therefore initiates an information-seeking subdialog.

In the general scheme, plans for achieving AAGs may be verbal or nonverbal. However, the only plans that concern us here are verbal, involving the production of an utterance via a speech act [Austin 1962]. The most straightforward form (and the only form used in this article) verbalizes a direct request to the user for information. A more subtle plan might ask a question whose answer could then be used to infer the desired information. More sophisticated plans can use indirect speech acts [Searle 1969] such as

(2c) S: Do you have a particular reason for going there?

Indirect speech acts are sometimes more courteous and also provide more flexibility; here, for example, (2c) does not presuppose the user to have a particular reason, whereas (2) requires the user to explicitly contradict the presupposition if it is unacceptable.

There are many potential reasons why an agent might wish to initiate a subdialog for the purpose of seeking information about the user. Fig. 1 shows a broad classification of some of these reasons. Our study targets only a high-level set of sources of AAGs, where the cause of an AAG stems from a failure in the system's plan recognition or planning processes.

The nature of AAGs can be understood in two different ways. In one view, acquisition goals in themselves are neither active nor passive, but the plans for

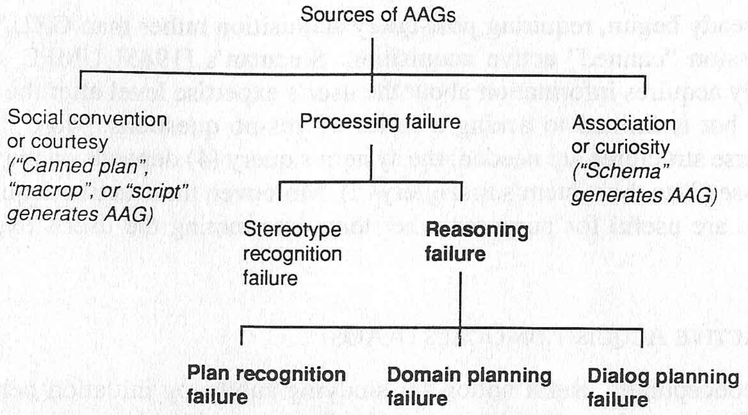
Sources of AAGs
Social convention or courtesy ("Canned plan", "macrop", or "script" generates AAG)
Processing failure
Association or curiosity ("Schema" generates AAG)
Stereotype recognition failure
Reasoning failure
Plan recognition failure
Domain planning failure
Dialog planning failure

Fig. 1. Potential sources of AAGs.

achieving them either involve active speech acts or passive inference. There are active acquisition *plans*, but AAGs are merely a conceptual shorthand for ordinary acquisition goals where our focus is on active means for satisfying them. In the second view, AAGs are seen as a real class of goals that delimit a narrower class of plans than ordinary acquisition goals. AAGs are directly indexed to plans involving active acquisition, speeding up memory retrieval for plans that satisfy AAGs. Flexibility is traded off for speed, because if the system adopts an AAG instead of a more general acquisition goal, it does not consider passive acquisition plans. This is an instance of a general tradeoff found in planners, where compiling frequently used plans can buy an overall speed-up at the price of misplanning in a small number of cases. The choice of perspective on the nature of AAGs depends on the investigator's goals. From the engineering standpoint, AAGs should index plans only if the overall speed-up outweighs the loss in flexibility. From the cognitive modeling standpoint, one would need psychological verification as to whether human dialog participants have active acquisition strategies compiled into conversational behavior. We remain agnostic on this issue in the absence of sufficient data of either sort.

2.2. GENERATING AAGS FROM REASONING FAILURES

Failures in the dialog system's normal reasoning processes are a highly productive source of AAGs, and account for dialogs like the example above. We assume that dialog processing requires three major kinds of normal reasoning processes: plan recognition, domain planning, and dialog planning. Briefly, AAGs can be produced by failures in the following manner:

Plan recognition: Plan recognition is one of the main passive acquisition strategies since it infers the user's underlying goals and plans so as to explain the user's speech acts. Moreover, the inferences produced by plan recognition are crucial to the cooperative continuation of a dialog. If the system is unable to construct any plan to explain the user's speech act, it might hypothesize one of two causes: (1) an unknown user goal, or (2) an unknown user misconception. In either case, it may generate an AAG to identify the unknown. If, on the other hand, the system is able to produce two or more plausible explanations for the speech act but cannot decide between them, it can generate an AAG to determine some fact that would eliminate all but one explanation.

Domain planning: To produce a solution for the user's goals (a route plan, in this article's examples), the system performs planning in its domain of expertise. When the system is unable to evaluate the expected utility of the plans it produces because of a lack of information about the user, an AAG to obtain the needed information can be generated. If the system is able to produce multiple plans of high expected utility but cannot choose between them, it can generate an AAG to determine the user's preference.

Dialog planning: The system performs dialog planning to determine its own course of action, particular with regard to its speech acts. In cases where the system is unable to evaluate the expected utility of potential alternative plans, it can generate an AAG to obtain relevant information.

The notion of expected utility is described subsequently in Section 3.2. Also, note that once an AAG is generated, the system may or may not actually adopt and attempt to satisfy the goal; this issue is also discussed later, in Section 3.1.

The following are a number of rules for generating AAGs under the failure conditions sketched above. The rules are summarized in Fig. 2 at the end of this section. This is a slightly updated version of the rules presented in an earlier paper [Wu & Horster 1989].

> *Rule 1a* If the plan recognizer produces no plan of acceptable utility explaining the user's speech act, then generate an AAG to identify an unknown user goal.

> *Rule 1b* If the plan recognizer produces no plan of acceptable utility explaining the user's speech act, then generate an AAG to identify (and/or correct) an unknown user misconception.

Rules 1a and 1b are responsible for (8a) or (8b) in the following example.

The system cannot produce a plan of acceptable utility to explain (7) and assumes either an unknown user goal or misconception. In (8a), an AAG to determine the unknown user goal is satisfied; (8b) corrects a possible user misconception.[1]

(1) U: How do I get to the center of the bay?
(2) S: Why do you want to go there?
(3) U: I want to take a picture of the skyline.
(4) S: Is it sufficient to drive to Treasure Island, or is it necessary to take a cruise?
(5) U: No, a cruise isn't necessary.
(6) S: Then you should drive to the Bay Bridge and take the Treasure Island exit.
(7) U: What about Angel Island?

(8a) S: Why do you ask?
(9a) U: I also want to visit Angel Island.

(8b) S: There is no bridge to Angel Island, you must take a ferry.

Rule 2 If multiple plans explain the user's speech act, but they are plans for independent user goals (so it makes no sense to compare their utilities), and all of the goals are plausible, then generate an AAG to disambiguate the user's intention.

Rule 2 is responsible for (4) in the following dialog. The system cannot determine whether the user's goal behind uttering (3) is to request additional directions or to convey a wish to eat in Chinatown, because either goal is plausible given the system's knowledge of the user's overall goals.

(1) U: Where can we eat a good Chinese meal tonight?
(2) S: There is a good restaurant called Mandarin House on Clement.
(3) U: Is that in Chinatown?

(4) S: No, do you want to eat in Chinatown?
(5a) U: Yes.
(6a) S: The Red Chamber on Jackson is quite good.

(4) S: No, do you want to eat in Chinatown?

---

[1] Rule 1b is just one case of suspecting a user misconception; detecting and correcting misconceptions is an orthogonal issue that many researchers have addressed independently [e.g., McCoy 1988, Chin 1988].

(5b) U: No, that doesn't matter.
(6b) S: To get to Clement, you can take Geary down to 30th and then go north one block.

Rule 3a If, while comparing the over-specificness of the highest-utility domain plans, no plan closely matches the specificness of the known user goals, and the plans have mutually exclusive consequences, then generate an AAG to determine a more specific user goal.

Rule 3a accounts for the following exchange:

(1) U: How do I get to the center of the bay?
(2) S: Why do you want to go there?

The system is able to build a number of plans, including driving to Treasure Island, chartering a yacht, windsurfing, and taking a bay cruise tour. Each plan's consequences are over-specific in comparison with the known goal of the user, which is to get to the center of the bay; thus, each plan strongly constrains the possible future actions. Moreover, each plan constrains future actions differently; one results in the user being on Treasure Island, another results in the user being on a boat, and so on. Since the user presumably has a particular purpose for getting to the center of the bay, the plan selected should be the one whose constraints most closely match his specific goal. However, since the system does not know the user's specific goal, it is unable to evaluate which plan's consequences provide the best match.

Rule 3b If, while comparing the difficulty of the highest-utility domain plans, some plans have a precondition that others do not, and the precondition is neither known to hold nor assumable by default, then generate an AAG to determine the ease of achieving the precondition.

Rule 3b produces (2) below, because the system needs to discover whether the user can satisfy the precondition of a plan, in order to select the easiest plan.

(1) U: How do I get to the Marina?
(2) S: Do you drive?

Rule 4 If the highest-utility domain plans are non-comparable, then generate an AAG to determine the user's preference.

Rule 4 is responsible for (4) in the following dialog, because the Treasure Island plan is superior to the cruise plan in terms of difficulty, but the cruise plan is superior in terms of being less susceptible to potential goal conflicts and resulting in a less over-specific state, because taking the picture from the water constrains the angle less than the island.

(1) U: How do I get to the center of the bay?
(2) S: Why do you want to go there?
(3) U: I want to take a picture of the skyline.
(4) S: Is it sufficient to drive to Treasure Island, or is it necessary to take a cruise?

Rule 5 If the dialog planner lacks information to compare the highest-utility dialog plans, then generate an AAG to obtain the needed information.

Rule 5 is illustrated by (2) below, where the system has built several different plans for expressing a location, but cannot judge how difficult to execute (or how likely to succeed) they are until it ascertains whether the user's knowledge satisfies the plans' preconditions.[2]

(1) U: Where is Cafe Rigoletto?
(2) S: Do you know where Symphony Hall is?
(3) U: Yes.
(4) S: Cafe Rigoletto is in the alley across the street from Symphony Hall.

3. Decision-Theoretic Implications

Plan recognition failure

- Plan recognizer produces no plan of acceptable utility explaining user's speech act

Rule 1a If the plan recognizer produces no plan of acceptable utility explaining the user's speech act, then generate an AAG to identify an unknown user goal.

Rule 1b If the plan recognizer produces no plan of acceptable utility explaining the user's speech act, then generate an AAG to identify (and/or correct) an unknown user misconception.

- Plan recognizer produces multiple plans of acceptable utility explaining user's speech act

Rule 2 If multiple plans explain the user's speech act, but they are plans for independent user goals (so it makes no sense to compare their utilities), and all of the goals are plausible, then generate an AAG to disambiguate the user's intention.

Domain planning failure

- Domain planner lacks information needed to compare plan utility

Rule 3a If, while comparing the over-specificness of the highest-utility domain plans, no plan closely matches the specificness of the known user goals, and the plans have mutually exclusive consequences, then generate an AAG to determine a more specific user goal.

Rule 3b If, while comparing the difficulty of the highest-utility domain plans, some plans have a precondition that others do not, and the precondition is neither known to hold nor assumable by default, then generate an AAG to determine the ease of achieving the precondition.

- Domain planner produces plans with non-comparable utility

Rule 4 If the highest-utility domain plans are non-comparable, then generate an AAG to determine the user's preference.

Dialog planning failure

- Dialog planner lacks information needed to compare plan utility

Rule 5 If the dialog planner lacks information to compare the highest-utility dialog plans, then generate an AAG to obtain the needed information.

Fig. 2. Taxonomy of rules for generating AAGs in response to failure conditions.

We believe that *decision-theoretic search* is the most appropriate framework for modeling the kind of planning and plan recognition needed for active acquisition. In this section, we examine several architectural requirements imposed by the active acquisition theory. These conditions are not met in most existing dialog systems, because they require the system to estimate the expected utilities of alternative plans, without actually executing them. This suggests using decision theory, which elegantly treats how to resolve choice points using statistically-derived heuristic estimates of expected utility. We identify five dimensions of expected utility that are relevant to active acquisition; a subtype of decision theory called multi-attribute utility theory is employed to handle the multiple dimensions of utility.

3.1. RATIONALITY REQUIREMENTS

Active acquisition requires that the following conditions on dialog system architecture hold:

(C1) *Rational* (non-arbitrary) *domain plan selection*. Rules 3a, 3b, and 4 generate AAGs from failures during the domain planning process, specifically, from the part of the process that selects between alternative plans when

---

[2] UMFE [Sleeman 1985] can be characterized as performing active acquisition in this class. The active acquisition queries are aimed at determining the user's expertise level, in order to formulate a felicitous response.

more than one is available. This assumes that the domain planner employs a rational strategy for selecting plans, based upon estimates of the expected utility of alternative plans. The rules could not be applied in dialog systems employing simpler planning models that arbitrarily select one plan when several options are possible.

(C2) *Rational dialog plan selection*. The same argument holds for dialog planning. Rule 5 generates AAGs from failures during the selection process in dialog planning, so the dialog planner must also employ a rational strategy for selecting plans.

(C3) *Rational plan recognition termination*. The system must be able to determine when the expected utility of attempting further plan recognition is lower than an alternative course of action. Rules 1a and 1b apply when "the plan recognizer produces no plan of acceptable utility explaining the user's speech act". In a realistically large domain, the search space—i.e., the number of explanations that could be produced, considered, then accepted or rejected—is enormous. This does not mean that exhaustive search must be performed before generating an AAG. Some heuristic means is needed to prune those search directions likely to produce only poor explanations, such as explanations involving unlikely user goals or superfluous operators. Moreover, for plan recognition problems that are likely to be time-consuming it may be quicker to take an interactive course of action such as requesting clarification from the user. Rules 1a and 1b, then, could be more precisely stated to apply "when it is pragmatic to terminate attempting to produce a plan of acceptable utility explaining the user's speech act". This is similar to Mayfield's [1991] principle of trading off between the difficulty and benefits of continuing a plan recognition search.

(C4) *Rational AAG plan selection*. The system must be able to compare the expected utility of alternate plans for different AAGs, in order to eliminate poorer AAG plans when they are superceded by better ones.

Unlike the earlier three conditions which related to generation of AAGs, condition (C4) relates to AAG selection. Multiple AAGs may be generated while processing a single user utterance, because we attempt to continue normal processing after a failure occurs as best as possible, collecting all AAGs. This will suggest a number of potential plans, some of which satisfy one AAG, and some of which simultaneously satisfy multiple AAGs. The system should select the plan that maximizes utility with respect to the system's overall goals of being cooperative, brief, comprehensible, and relevant [Grice 1975]. Consider the following dialog:

(1) U: Do you know good restaurants in this neighborhood?
(2) S: Yes, are you interested in a particular style of cuisine?

(2) is the result of a plan that simultaneously satisfies two different AAGs generated while processing (1). One AAG, produced by Rule 2, is to determine whether the user's goal in asking (1) is to discover the system's breadth of knowledge or to discover good restaurants. The other AAG, produced by Rule 3a under the assumption that the user's goal is to discover good restaurants, is to determine information needed to compare the utility of various restaurant-going plans (i.e., domain plans). (2) is a better response than either "Yes, do you want to know some?", which satisfies the former AAG, or "What style of cuisine?", which satisfies the latter.

The system must also compare plans for AAGs against other dialog plans not involving AAGs, since active acquisition is not always the best response when a reasoning failure occurs. One example is Rule 1b, where the system has the option of correcting a suspected user misconception without actually determining whether it exists. The following example demonstrates another case:

(1) U: Where can I buy a newspaper?
(2) S: In the tobacco shop across the street.

Suppose that there is actually a newspaper vending machine a few steps closer than the tobacco shop, but it only sells the Chronicle and moreover requires exact change. There is not enough information to compare the plan involving the vending machine with the tobacco shop plan, so Rule 3b generates an AAG to determine the ease of meeting the preconditions of the vending machine plan. However, the time needed to walk the few extra steps to the tobacco shop is less than that required by an active-acquisition subdialog. Since the system's overall goal is to be as cooperative with the user as possible, it decides not to satisfy the AAG and chooses the non-active-acquisition plan instead, producing (2).

3.2. MULTI-ATTRIBUTE UTILITY

Conditions (C2), (C3), and (C4) all require the system to evaluate the expected utility of its alternative courses of action so that it can execute the most promising. Condition (C1) involves choosing between hypothetical domain plans for the user; even though the system is not itself contemplating executing these plans, it is reasonable to use the same utility evaluation functions. Condition (C3) has the peculiar characteristic that it treats "thinking", i.e., plan recognition computation, as an action in itself (one way of handling this will be described below).

We have already alluded to several types of utility considerations: difficulty of plan execution, degree of conflict with other goals, and over-specificness

	AAG generation			
	Domain plan selection failure	Dialog plan selection failure	Plan recognition termination	AAG selection
(1) Difficulty	Y	Y	Y	Y
(2) Degree of goal conflict	Y	Y		Y
(3) Over-specificness	Y			
(4) Productiveness	?	?	Y	Y
(5) Likelihood of success	?	Y	Y	Y

Fig. 3. Usefulness of individual attributes for AAG generation and selection.

of the result state. In general, dialog agents must consider many dimensions of utility when comparing plans. Thus, a *multi-attribute utility* metric is most appropriate, where each plan is assigned a vector of values for the different utility attributes.[3] A multi-attribute utility metric establishes a partial order on a set of plans, such that a plan is judged to be better than another plan if and only if it is superior in all attributes. Two plans are *non-comparable* if neither plan is superior in all attributes.

A major issue in decision-theoretic models is the estimation of expected utility. We now propose five utility attributes for active acquisition and their estimation methods. Attributes (1)–(3) are required because AAGs are generated from rational plan selection failures involving the estimation or comparison of these attributes. We suggest attributes (4)–(5) as the minimal set of attributes needed for decision-making involving rational plan recognition termination and AAG selection; in reality, additional attributes will probably be desirable to further increase efficiency and plan discrimination.[4] Fig. 3 shows more precisely how we envision the various attributes' usefulness.

(1) The *difficulty* attribute measures the effort or time required of the individual steps of the plan, as well as the difficulty of achieving the preconditions (if they are not already satisfied). It can be estimated statistically by summing average execution times (or other cost functions such as effort or energy) for the individual steps of the plan.

(2) The *degree of goal conflict* attribute measures how much unintended side-effect consequences of a plan conflict with other goals. It can be computed as a weighted count of conflicting goals, where the weights reflect the relative importance assigned to each goal.

(3) The *over-specificness* attribute is more difficult to define. We would like some metric that describes how much more specific the plan's consequence description is than the goal description. Ideally, by viewing the goal

---

[3] See von Winterfeldt & Edwards [1986] for an introduction to multi-attribute utility theory.

[4] In particular, a *risk* attribute is used in many decision-theoretic applications and most likely ought to be employed in dialog systems as well. Its omission here is purely methodological: a risk attribute was not required to explain active acquisition in any of the examples in our study, and evaluating expected risk is non-trivial.

description as a set of states in a state space representing the real world, and the plan consequence description as a more specific subset of those states, we can view the over-specificness attribute as the ratio of the sizes of the state sets. In other words, we would like to determine what percentage of states matching the goal description also match the plan consequence description. Unfortunately, it makes little sense to count states in the planner's representation language, because the choice of states in the representation is arbitrary and depends entirely on the designer. What can be done instead is to attach numeric weights to each state description in the knowledge base, as an approximate measure of the size of the corresponding real-world state sets. For example, if the "being at the Marina" state is arbitrarily assigned a weight of 100, the more specific "being at the foot of the Marina pier" state might be assigned a weight of 60, indicating that within the system's experience, 60% of the "being at the Marina" states were eventually resolved to "being at the foot of the Marina pier".

(4) The *productiveness* attribute measures how much a plan achieves. Productiveness can be computed by counting the number of goals achieved by the plan, or, more precisely, by summing the importance weights of the goals achieved by the plan.[5] As an example of how the productiveness attribute helps facilitate rational AAG selection, again consider the dialog:

(1) U: Do you know good restaurants in this neighborhood?
(2) S: Yes, are you interested in a particular style of cuisine?

The plan that produces (2) is chosen over the less productive plans that produce "Yes, do you want to know some?" and "What style of cuisine?" which satisfy only one of two AAGs.

(5) The *likelihood of success* attribute measures how likely the plan is to succeed. Estimation is based on the frequency of unexpected failure in previous experience.

The estimation techniques for difficulty, over-specificness, and likelihood of success are all based on prior experience and thus require statistical data, data that is not likely to be readily available. The theoretical approach to gathering the numbers is to apply an incremental learning algorithm to some training corpus of dialogs. Research on learning techniques being at a relatively immature stage, this may not be practical. Such an approach would also require a sizeable training corpus, which either the system would have to be capable by itself of analyzing—parsing, interpreting, segmenting—or would need to be extensively annotated by hand. Moreover, not very much

---

[5] Choosing the plan with the highest productiveness is a general rule that subsumes two principles for meta-planning ("meta-themes") described by Wilensky [1983, p. 31]: "achieve as many goals as possible", and "maximize the value of the goals achieved".

statistical data on difficulty, over-specificness, and likelihood of success for *domain* plans can be acquired from consultation transcripts; however, this problem applies to all general planning models and is not specific to active acquisition.

A more engineering-oriented approach is to guess at the numbers by trial and error until they produce something close enough to the desired results. Optimal results are not likely to be achieved by number twiddling. However, it should be possible to meet the most important criteria: that the system err on the conservative side, i.e., asks a potentially unnecessary clarification question rather than failing to volunteer a potentially relevant plan. Another tactic is to assign default "no information" null values to utility attributes for which insufficient knowledge engineering is available. Attempting to evaluate or compare null-valued attributes can either be made to produce random results (which is no worse than existing systems) or generate failures (which might require additional failure-handling AAG rules). So long as utility attributes are treated as a means of pruning only *obviously* poor plans, some performance gain should be achievable without undesirable side-effects.

A possible compromise is to combine the approaches. For example, difficulty values for dialog planning may be feasible to collect using learning techniques since dialog planning is a fairly restricted domain, whereas values for more general domains such as route planning may have to be estimated, at least until better learning methods are developed for general planners.

4. Architecture

The decision-theoretic approach requires a reorganization of conventional dialog architectures. Fig. 4 shows the flow of information for the generation of AAGs in MAPS, a preliminary implementation we are developing. If the division of functions is viewed abstractly, the information flow requirements are applicable to most current dialog architectures. Both domain planning and dialog planning must consult the utility evaluation and comparison functions. Plan recognition can also use the utility functions to rank its alternative explanations of user speech acts, though we have not yet explored minimal pairs demonstrating necessity of this. The functions that handle failures are responsible for generating AAGs, which must then be fed back into the dialog planner's set of candidate goals. There, the AAGs must be weighed against other possible non-active-acquisition courses of action.

Active acquisition demands a more flexible control strategy than in pipeline architectures, yet a more centrally-planned control strategy than in blackboard architectures. A pipeline architecture does not permit alternating control between plan recognition, domain planning, and dialog planning. On the other hand, a blackboard architecture, by running all processes whenever possible, does not allocate computational resources as wisely as a centralized control mechanism can. The architecture of MAPS implements the information flow described above using a centralized scheduler that takes advantage of the decision-theoretic expected utilities that are already needed for AAG generation. In essence, the utility estimates are used to guide navigation through the search space representing possible courses of action. The architecture incorporates the following features, depicted graphically in Fig. 5:

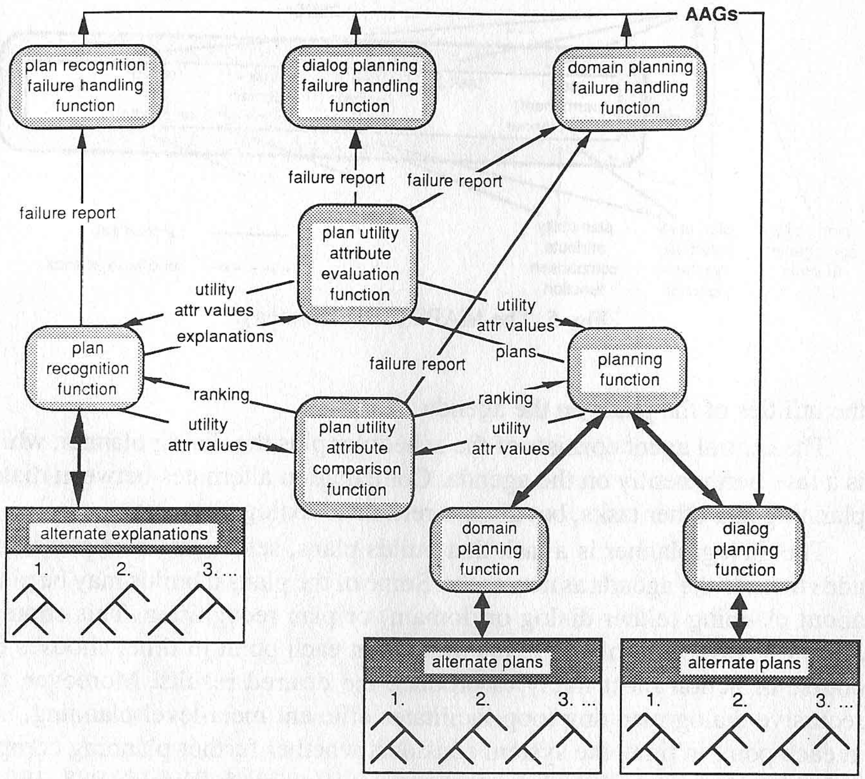

Fig. 4. Flow of information for active acquisition.

(1) A central planning agent, that builds and initiates execution of plans, thus determining the dialog system's actions [Chin 1988].

(2) Primitive functions for estimation and comparison of plan utilities.

(3) An *agenda* of tasks, each of which represents the execution state of a partially-executed plan.

(4) An *aggregation function* for converting utility estimates into task priorities.

(5) A *scheduler* that executes and suspends tasks, periodically reviewing

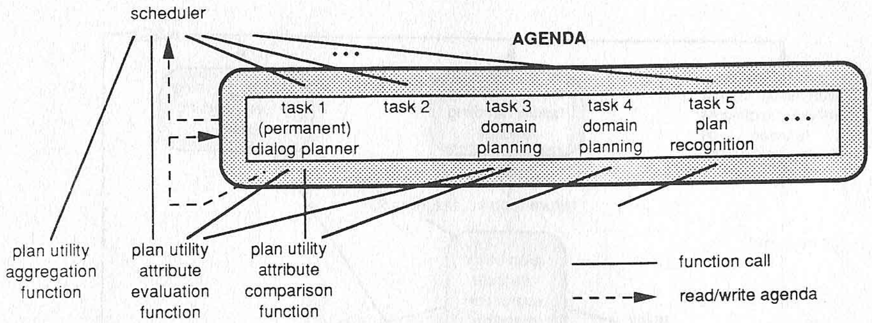

Fig. 5. The MAPS control strategy.

the utilities of the plans on the agenda.

The central agent consists of the scheduler plus the dialog planner, which is a task permanently on the agenda. Computation alternates between dialog planning and other tasks, but always returns to dialog planning.

The dialog planner is a task that builds plans, selects the best plans, and adds them to the agenda as new tasks. Some of the plans it builds may be plans about planning (either dialog or domain) or plan recognition. This strategy allows flexible control where the system, at each point in time, chooses the course of action most likely to produce the desired results. Moreover, the recursive dialog planning loop facilitates efficient meta-level planning, i.e., at each point in time, the system considers whether further planning computation is more worthwhile than other actions [Russell & Wefald 1988, 1991]. Although we have not yet found meta-planning to be absolutely necessary in our active acquisition theory, it seems wise to employ a multi-level architecture with general meta-reasoning facilities. We are also studying cases where AAGs are produced by failures in meta-planning.

The purpose of the aggregation function is to translate a vector of utility attribute values into a scalar priority value, thus allowing the scheduler to select the highest-priority task. The most straightforward, *additive* aggregation function is simply a weighted sum of the attribute values. Compressing a partial ordering into a full ordering, though necessary, introduces some arbitrariness that should be minimized. In particular, it is worth noting that attributes are often not entirely independent (though this makes no difference in how we treat them elsewhere in the system). For example, there is clearly some interaction between productiveness and degree of goal conflict, since any goal conflict limits the productiveness of a plan. Similarly, difficulty and likelihood of success in most domains stand in a loosely inverse relationship. When attributes are (partially) correlated in this way, the weight given to those attributes must compensate for the fact that a change in any one attribute value will show up in another attribute as well.

The functions for evaluating, comparing, and aggregating plan utilities are assumed to be primitive. It is particularly important that the evaluation and aggregation functions incur negligible execution cost, to avoid inadvertently distorting the expected utility values on which scheduling decisions are based. If any of these functions require significant computational resources, their costs and benefits should also be taken into account in making control decisions.[6]

5. Example

The following trace of an example interchange demonstrates the effect of several AAG rules. For the initial implementation, stubs are being used in place of an actual parser, analyzer, and generator; however, they are assumed to operate just as the components in existing dialog architectures such as UC [Wilensky et al. 1988].

```
Executing listen

User>  How do I get to the center of the bay?

Executing cooperatively-handle-user-speech-act
Executing recognize-user-speech-act
Executing analyze-current-user-utterance
Executing extract-user-plans

Extracted user plans:
  user-has-plan (plan1)
  plan-has-goal (plan1, goal1)
  user-knows-route-goal (goal1, location1, location2)
  here (location1)
  center-of (location2, bay1)
  san-francisco-bay (bay1)
  plan-composition (plan1, query-system1)
```

The extracted user plan is the relevant output from the plan recognizer. The symbol `query-system1` is the root node of a structure representing the user's plan to learn the route by querying the system. The system now identifies the goal it should adopt in order to help the user, and then begins domain planning.

```
Executing identify-cooperative-goals

Identified cooperative goal:
```

---

[6] This requires an extra meta-level in the search structure, to mediate computation resources between the scheduler (including the utility functions it calls) and agenda tasks.

```
  user-knows-route (location1, location2)
  center-of (location2, bay1)
  san-francisco-bay (bay1)

Executing plan-route
Executing plan-for-goal
Executing plan-for-goals
Executing collect-plans-for-goals
Executing select-best-plan-for-context

Failure detected in plan utility attribute comparison function
  ... no plan closely matches specificness of known user goals
Over-specificness of drive-to-treasure-island = 1/0.15 = 6.66
Over-specificness of embark-on-bay-cruise-tour = 1/0.10 = 10.00
Over-specificness of embark-on-bart-through-tube = 1/0.20 = 5.00
  ... some plans have mutually exclusive consequences
Consequences of drive-to-treasure-island:
           be-at-treasure-island
Consequences of embark-on-bay-cruise-tour:
           be-on-path-in-bay-with-view-of-sf
Consequences of embark-on-bart-through-tube:
           be-in-bay-under-water
Activating AAG Rule 3a
  ... generating AAG: determine-more-specific-user-goal
Adopting AAG: determine-more-specific-user-goal
```

The system detects that it lacks the information needed to compare the three route plans it has produced. All three are excessively over-specific, and the plan's consequences exclude each other. The over-specificness measure used here is the inverse of the (hand-estimated) percentage of occurrence in an expert's experience of the subclass within the superclass. For example, driving to Treasure Island is estimated to account for 15situations in middle of the bay, yielding an over-specificness measure of 1/0.15. The system now attempts to satisfy the AAG generated by Rule 3a.

```
Executing query-user-for-more-specific-goal
Executing express-request-for-more-specific-user-goal

System> Why do you want to go there?

Executing listen

User> I want to take a picture of the skyline.

Executing recognize-user-speech-act
Executing analyze-current-user-utterance
Executing extract-more-specific-user-goal

Extracted user goal:
  user-has-goal (picture-taking1)
  event-located-at (picture-taking1, location2)
```

```
  center-of (location2, bay1)
  user-subplan-for (picture-taking1, going1)
  event-path-through (going1, location1, location2)
  user-prerequisite-for (going1, knowing-route1)
  user-has-goal (knowing-route1)
  here (location1)

Resuming select-best-plan-for-context
```

Having determined the user's goal more specifically, the system now attempts again to choose the best route plan. However, it discovers that neither of the two top-ranked plans are superior in all attributes. Driving to Treasure Island is less difficult than taking a bay cruise (the difficulty values are computed as a sum of the difficulty values of all preconditions and plan steps, again hand- estimated at some base level), but Treasure Island is now more over-specific because the context is the user's skyline-photo goal. The system therefore adopts the AAG generated by Rule 4, and asks the user for a preference.

```
Failure detected in plan utility attribute comparison function
  ... highest-utility plans are non-comparable
Difficulty of drive-to-treasure-island = .31
Difficulty of embark-on-bay-cruise-tour = .46
Over-specificness of drive-to-treasure-island = 1/0.25 = 4.00
Over-specificness of embark-on-bay-cruise-tour = 1/0.65 = 1.54
Activating AAG Rule 4
  ... generating AAG: determine-user-preference
Adopting AAG: determine-user-preference

Executing query-user-for-preference
Executing express-request-for-user-preference

System> Is it sufficient to drive to Treasure Island,
                  or is it necessary to take a cruise?

Executing listen

User> No, a cruise isn't necessary.

Executing recognize-user-speech-act
Executing analyze-current-user-utterance
Executing extract-user-preference
Resuming select-best-plan-for-context
Executing express-route-plan

System> Then you should drive to the Bay Bridge and
                    take the Treasure Island exit.

Executing listen
```

Once the original user query has been satisfactorily answered, the system returns to a listening state. If the user's subsequent question cannot be successfully analyzed by the plan recognizer, Rule 1 generates another pair of AAGs.

```
User>  What about Angel Island?

Executing cooperatively-handle-user-speech-act
Executing recognize-user-speech-act
Executing analyze-current-user-utterance
Executing extract-user-plans

Failure detected in plan recognition
  ... no plan explaining user's speech act
Activating AAG Rule 1a
  ... generating AAG identify-unknown-user-goal
Activating AAG Rule 1b
  ... generating AAG identify-unknown-user-misconception
Adopting AAG identify-unknown-user-goal

Executing query-user-for-unknown-goal
Executing express-request-for-unknown-goal

System>  Why do you ask?

Executing listen

User>  I also want to vist Angel Island.
```

6. Conclusion

Our analysis of active acquisition applies the decision-theoretic principle of maximizing *expected utility* to dialog systems in two ways: (1) it is assumed that the system performs planning and plan recognition in order to determine the course of action that maximizes its expected utility, and that failures can occur when reasoning about expected utilities, and (2) it is assumed that after a failure occurs, the system selects the recovery option that maximizes expected utility, which may (or may not) require satisfying an AAG.

We have examined theoretical requirements of active acquisition that are not met in the planning and plan recognition models used by existing dialog systems. Multi-attribute utilities will be required for any feasible implementation of active acquisition. We then described a control architecture that supports multi-attribute decision-theoretic planning and execution.

The long-term goal of this work is to construct a performance theory of how and when a dialog agent should actively acquire information for the user model. We feel that decision-theoretic active acquisition is the natural synthesis of plan-based speech act theories with dialog agent models. As plan recognition models evolve to handle broader types of ill-formed input (e.g., Eller & Carberry [1991]), they hypothesize increasing numbers of alternative explanations, requiring the dialog agent to choose from a greater number of possible reactions. Further avenues of investigation include adding other utility attributes and refining the estimation methods. The coherence and information content metrics used by Raskutti & Zukerman [1991] to rank and prune alternative explanations can be viewed as additional utility attributes (or subcomponents of existing attributes such as likelihood of success and productiveness). Plan recognizers that assign probabilities to alternative explanations [Calistri-Yeh 1991, Raskutti & Zukerman 1991] can be used to increase utility estimation accuracy when the user's true goals cannot be determined with certainty, by using probabilities as weights for computing weighted averages of utility attribute values.

Acknowledgements

Many thanks to Bettina Horster, my collaborator in the development of the active acquisition theory. The comments of an anonymous referee were very helpful. I would also like to thank Robert Wilensky and the members of the BAIR (Berkeley Artificial Intelligence Research) group.

References

Allen, James F. and Raymond C. Perrault: 1980, 'Analyzing intention in utterances'. *Artificial Intelligence* **15**: 143–178.

Austin, J. L.: 1962, *How to do things with words*. London: Oxford University Press.

Calistri-Yeh, Randall J: 1991, 'Utilizing user models to handle ambiguity and misconceptions in robust plan recognition'. *UMUAI*, forthcoming.

Carberry, Sandra: 1988, 'Modeling the user's plans and goals'. *Computational Linguistics* **14**(3): 23–37.

Chin, David: 1988, *Intelligent agents as a basis for natural language interfaces* (Ph.D. Thesis). Comp. Sci. Div., Univ. of California at Berkeley, Report No. UCB/CSD 88/396.

Cohen, Philip R., and C. Raymond Perrault: 1979, 'Elements of a plan-based theory of speech acts'. *Cognitive Science* **3**: 177–212.

Eller, Rhonda, and Sandra Carberry: 1991, 'A meta-rule approach to flexible plan recognition in dialogue'. *UMUAI*, forthcoming.

Grice, H. P: 1975, 'Logic and conversation'. In P. Cole and J. L. Morgan (eds.), *Syntax and Semantics, Volume 3*. New York: Academic Press.

Kobsa, Alfred: 1985, 'VIE-DPM: A user model in a natural-language dialogue system'. In J. Laubsch (ed.), *GWAI-84, 8th German Workshop on Artificial Intelligence*. Berlin: Springer-Verlag.

Litman, Diane J., and James F. Allen: 1984, *A plan recognition model for subdialogues in conversation.* Dept. of Comp. Sci., Univ. of Rochester, TR 141.

Mayfield, James: 1989, *Goal analysis: Plan recognition in dialogue systems* (Ph.D. thesis). Comp. Sci. Div., Univ. of California at Berkeley, Report No. UCB 89/521.

Mayfield, James: 1991, 'Controlling inference in plan recognition'. *UMUAI*, forthcoming.

Raskutti, Bhavani, and Ingrid Zukerman: 1991, 'Generation and selection of likely interpretations during plan recognition in task-oriented consultation systems'. *UMUAI*, forthcoming.

Rich, Elaine: 1979, *Building and exploiting user models*. Ph.D. Thesis, Comp. Sci. Dept., Carnegie-Mellon Univ.

Russell, Stuart, and Eric Wefald: 1988, 'Multi-level decision-theoretic search'. *AAAI Symposium on Game-Playing*, Stanford Univ.

Russell, Stuart, and Eric Wefald: 1991, 'Do the right thing'. Cambridge, MA: MIT Press.

Searle, J. R: 1969, *Speech acts: An essay in the philosophy of language*. Cambridge: Cambridge University Press.

Sleeman, D: 1985, 'UMFE: A user modelling front-end subsystem'. *International Journal of Man-Machine Studies* **23**: 71–88.

von Winterfeldt, Detlof, and Ward Edwards: 1986, *Decision analysis and behavioral research*. Cambridge: Cambridge University Press.

Wahlster, W., H. Marburger, A. Jameson, and S. Busemann: 1983, 'Over-answering yes-no-questions: Extended responses in a NL interface to a vision system'. *Proc. IJCAI-83*, 643–646.

Wahlster, W., and A. Kobsa: 1986, *Dialog-based user models*. Dept. of Comp. Sci., Universitaet des Saarlandes, West Germany, XTRA Report No. 3.

Wilensky, Robert: 1983, *Planning and understanding*. Reading, MA: Addison-Wesley.

Wilensky, Robert, David Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu: 1988, 'The Berkeley UNIX Consultant project'. *Computational Linguistics* **14**(4): 35–84.

Wu, Dekai, and Bettina Horster: 1989, 'Active acquisition for user modeling in dialog systems'. *Proc. Cognitive Science 1989*. Hillsdale, NJ: Lawrence Erlbaum Associates.