Evaluating Plan Recognition Systems: Three Properties of a Good Explanation

James Mayfield
The Johns Hopkins University Applied Physics Laboratory

(Received ..... ; Accepted in final form .....)

Abstract. Plan recognition in a dialogue system is the process of explaining why an utterance was made, in terms of the plans and goals that its speaker was pursuing in making the utterance. I present a theory of how such an explanation of an utterance may be judged as to its merits as an explanation. I propose three criteria for making such judgments: applicability, grounding, and completeness. The first criterion is the *applicability* of the explanation to the needs of the system that will use it. The second criterion is the *grounding* of the explanation in what is already known of the speaker and of the dialogue. Finally, the third criterion is the *completeness* of the explanation's coverage of the goals that motivated the production of the utterance. An explanation of an utterance is a good explanation of that utterance to the extent that it meets these three criteria. In addition to forming the basis of a method for evaluating the merit of an explanation, these criteria are useful in designing and evaluating a plan recognition algorithm and its associated knowledge base.

Key words: Plan recognition, explanation, evaluation, natural language processing, dialogue systems

In its broadest interpretation, an *explanation* of an event is a set of conditions that allow or cause the event to occur. There is a huge number of such explanatory conditions for any given event. To be useful in understanding or in formulating a response to an observed event, an explanation of that event must contain only a subset of these conditions. Some of the more important types of conditions are:

1. States of the world

2. Causal relationships

3. Beliefs, intentions, and affect of agents

4. Other events, including actions

5. Physical laws

The types of conditions selected to be part of an explanation dictate the character of the explanation. For example, the following are potential explanations for the overwriting of a particular file:

1. The Emacs editor overwrote it
2. Melissa edited it
3. Melissa was feeling malicious
4. It wasn't write-protected

At first, this final statement seems to be less an *explanation* of the event than an enabling condition of it. However, it seems to be a perfectly good explanation of the event relative to the question 'How could it have been overwritten?' Other types of explanations include scientific explanations, political explanations, affective explanations, *etc.* Schank (1986) even suggests explanations composed of questions.

These examples demonstrate that there are many different types of explanations for a single event that might be useful for a particular task. This paper details what it means for an explanation of an utterance (which is one kind of event) to be a good one. I formalize the concept of 'goodness' by suggesting three criteria by which the quality of an explanation of an utterance may be judged. While these criteria are intended to apply to explanations of *utterances*, they are also useful for a broad range of explanation-based tasks.

There are three reasons that it is useful to study criteria for evaluating explanations. First, they can be of help in the design of a knowledge base, because they indicate what types of representations will lead to the desired explanations. Secondly, they assist in the design of a plan recognition algorithm by placing constraints on the output of such an algorithm. Finally, such criteria provide a means to evaluate the output of a plan recognizer, once it has been implemented.

To demonstrate the usefulness of these criteria in the evaluation of plan recognition systems, I apply them to the output of the plan recognizer for the Unix Consultant (UC) project (Wilensky *et al.*, 1984; Wilensky *et al.*, 1988). UC serves as a consultant for novice users of the UNIX operating system. UC's plan recognition component is called PAGAN (for Plan And Goal ANalyzer). PAGAN's job is to build explanations of the user's utterances.[1] That is, PAGAN must infer an interrelated set of plans and goals held by the user that led her to make the utterance. The representation of plans and goals that PAGAN uses,

[1] This is not the same as explaining a concept to the user. The creation of this latter sort of explanation is an act of generation rather than one of understanding, and is performed in UC by the planner and the language generator.

as well as its algorithm for constructing explanations, are described in Section 5.

1. Criteria for evaluating explanations

A good explanation of an utterance meets the following criteria:

1. Applicability
2. Grounding
3. Completeness

The applicability criterion states that a good explanation of an utterance is applicable to the needs of the system that will use that explanation. The grounding criterion states that a good explanation of an utterance is grounded in what is already known of the speaker and of the dialogue. The completeness criterion states that a good explanation of an utterance covers every aspect of the utterance in depth; it leaves no portion of the utterance unexplained. These criteria are valid independent of the algorithm used to construct the explanation. That is, they are criteria that one applies to an explanation to determine how good it is, not to an algorithm to see how well the algorithm creates explanations (although they can of course be applied to the *output* of a plan recognition algorithm). In the following sections, I describe each of these criteria in more detail.

2. The principle of applicability

The principle of *applicability* states that a good explanation of an utterance is applicable to the needs of the system that will use it. That is, no matter how good an explanation might be in other respects, if it doesn't give the system that will use it what that system needs to do its job, then it is not a good explanation.

An assumption that underlies the principle of applicability is that a system that is trying to explain an utterance has interests of its own, which understanding the utterance might help to further. Thus, this principle implies that the operation of a plan recognizer cannot be independent of the system in which it is embedded, and therefore the concept of a domain-independent plan recognizer is impractical. Of course, it might be possible to isolate the portions of a plan recognizer that rely on the particular task to be performed, and represent those portions declaratively. The remaining inference engine would be domain-independent, but the plan recognizer as a whole would not.

The reader might have noticed that I have not included *accuracy* among the criteria for evaluating explanations. There are two reasons for this. First, accuracy is in general subsumed by applicability. That is, in most cases an explanation of an utterance that does not accurately reflect the speaker's intentions will not be applicable to the needs of the system. Secondly, in some cases, inaccurate explanations are perfectly acceptable. A system designed to simulate a paranoid schizophrenic for example might prefer a delusional interpretation of an utterance to an accurate one.

Closely related to the issue of accuracy is the issue of user misconceptions. It is possible for the user's beliefs to be at odds with the system's beliefs. In such cases, the plan recognizer should usually faithfully model the user's beliefs, even when they are incorrect. Much research has been devoted to the recognition of misconceptions, including that of Calistri-Yeh (1991), Chin (1988), Eller and Carberry (1992), Pollack (1984), Quilici (1989), and Retz-Schmidt (1991).

There are three dimensions of an explanation along which applicability may be assessed:

1. Composition
2. Content
3. Granularity

Applicability of composition concerns the *type* of element out of which explanations should be constructed. Applicability of content deals with the particular choice of elements that compose an explanation. Finally, applicability of granularity covers the level of generality of the elements of a particular explanation. These dimensions are discussed in the following sections.

2.1. Applicability of composition

The principle of *applicability of composition* holds that the *type* of the elements that compose an explanation must be applicable to the needs of the system. The composition of a good explanation is largely a reflection of how that explanation will be put to use; different tasks require different types of explanations. For example, a system whose task is to build a model of a user's knowledge will need to build explanations that are composed of facts about the user that allowed the user to produce an utterance. In such a system, an explanation of:

User: Can you tell me how to delete a file?

might be composed of facts such as:

1. The user does not know how to delete a file.

2. The user believes that the system knows how to delete a file.

3. The user believes that the system will cooperate with the plan.

4. The user knows what a file is, and what it means to delete one.

5. The user understands the task of the system.

6. The user understands English.

For a consulting system such as UC, the knowledge that is applicable to the system's task is knowledge of the plans and goals of the user. This is because the purpose of such a system is to address the user's goals. Thus, an explanation of this question for UC's purposes might be composed of facts such as:

1. The user wants to know how to delete a file.

2. The user expects to be told how to delete a file.

3. The user wants to delete a file.

4. The user wants to release disk space.

Notice that these facts include both goals that the speaker holds and actions that the speaker expects the hearer to carry out. Because the understanding of a speaker's plans and goals is crucial to so many domains that might benefit from a natural-language interface, this paper focuses on explanations composed of plans and goals.

2.2. Applicability of content

The principle of *applicability of content* holds that each of the components of a particular explanation should be applicable to the needs of the system. A typical utterance is rarely aimed at achieving a single isolated goal, but rather sits at the tip of a whole chain of motivating goals. Not all such goals will be applicable to a given system though. A good explanation of an utterance will not include all of these goals, but will include only those that are applicable to the purposes of the system. For example, the novice who asks:

User: How can I print a file on the laser printer?

might have asked the question so as to find out how to get a printout, so as to obtain a printout of the file, so as to check its contents for accuracy, so as to turn in an accurate report, so as to get a good grade, so as to graduate with a high grade point average, so as to get a good job, and so on. Only a portion of this chain of motivating goals is likely to be applicable to the purposes of a particular system, and a good explanation will include only that portion. UC's contract with its users for example is to provide information on the use of the UNIX operating system. Thus, in the UC context, a good explanation will not include goals that go beyond the use of UNIX. In this example, the last few goals in the list of motivating goals are beyond UC's purview, and therefore should not be a part of an explanation designed for use by UC.

At the other end of the spectrum, a good explanation of an utterance must include at least one goal that is within the system's domain of expertise. Thus, if an explanation of the utterance:

> USER: I have a problem with ls.

includes only the goal of informing the hearer of the problem or the goal of the hearer knowing about the problem, that explanation is an inadequate one for the UC domain. An adequate explanation of this statement for UC's purposes would indicate that the user is attempting to solicit help from UC, since this is a goal that is applicable to UC's task. In practice, adherence to the principle of completeness (described in Section 4) ensures that an explanation will have *at least* this desired depth.

2.3. Applicability of granularity

The third dimension along which applicability can be assessed is the granularity of the explanation. The principle of *applicability of granularity* holds that a good explanation contains the right amount of detail. It is often possible to divide a single action into a number of subactions. The extent to which an explanation is divided in this way is its level of granularity. Different levels of granularity may be more or less applicable to a given system. For example, suppose a user tells UC:

> USER: I want to add public read permission to my file called rastelli.

An explanation of this statement that indicates that speaker probably wants to use a UNIX command has a reasonable level of granularity for UC. On the other hand, an explanation that states that the speaker probably wants to type each letter of the name of a UNIX command and each letter of the file name is not a reasonable one for UC's purposes. While it is likely to be correct, this explanation is nevertheless too fine-grained to be of use.

Of course, if UC is concerned that the user might not know how to spell the command name, then the latter explanation above may be quite applicable. This points out that the type of explanation that is applicable can change as the short-term goals of the system change. Thus, such goals must always be taken into account when judging the merits of an explanation.

3. The principle of grounding

The second criterion for evaluating an explanation is the principle of *grounding*. This principle states that a good explanation of an utterance relates what is inferred from that utterance to what is already known about the speaker and the dialogue. Typically, before hearing an utterance, a system will already have some knowledge that is applicable to the processing of that utterance. This knowledge is of two types:

1. Knowledge of the dialogue

2. Knowledge of the speaker

First, the system might have already engaged in dialogue with the speaker. Secondly, the system might have some stored knowledge about the user, or be able to make informed guesses about what the speaker knows or believes.

It follows from the existence of these two kinds of knowledge that there are two ways that an explanation can be grounded in existing knowledge:

1. By relating it to knowledge of the dialogue

2. By relating it to knowledge of the speaker

The first way that an explanation can be grounded in existing knowledge is by relating it to knowledge of the dialogue. For example, it is difficult to imagine a good explanation for the utterance:

USER: It's called crazy-fingers.

that does not relate the utterance to some preceding dialogue. The main reason for this is that the subject of the sentence is a pronoun with no clear referent. Consider now the exchange:

UC: What's the name of the file?

USER: It's called crazy-fingers.

In this example, UC's question provides a background against which the user's statement can be understood. A good explanation of the user's utterance must relate the utterance to this previous dialogue.

A second way that an explanation can be grounded in existing knowledge is by lending credence to, or by casting doubt on, something that the system already believes of the user. In this example, the user's response in the exchange:

UC: What's the name of the file?

USER: It's called crazy-fingers.

might lend credence to UC's belief that the user knows the 'ls' command (which lists file names in UNIX), thereby grounding the statement in knowledge that UC previously held about the speaker (in addition to grounding it in UC's knowledge of the dialogue). This kind of grounding lies in areas of user modeling outside of plan recognition, and I will not discuss it further. See Kass and Finin (1988) for an introduction to the broad range of components of a general user model.

An assumption that motivates the principle of grounding is that the utterance to be explained is part of a larger rhetorical structure such as a dialogue or text. The significance of this assumption is that interactive systems that don't engage in dialogue, such as simple question-answering systems, can largely ignore the principle of grounding. However, such systems are limited in their usefulness as natural language systems; dialogue is an important part of language.

There are two important aspects of an explanation that is grounded in knowledge of preceding dialogue:

1. The type of concept to which the explanation is attached

2. The type of attachment

First, it is useful to categorize the type of knowledge to which the utterance is connected. There are many such concepts; I am concerned only with plans and goals. An explanation may attach to either of these components of the dialogue model, or possibly to both of them. Secondly, the type of attachment to a concept is important, and can be categorized. The following sections are divided according to the type of knowledge to which an explanation is connected. Within each section, I discuss the ways that an explanation can be attached to a concept of that type.

3.1. How an explanation attaches to a plan

There are three ways that an utterance can connect to a plan that a speaker is already pursuing:

1. Plan continuation
2. Plan delay
3. Plan rejection

These types of grounding reflect the status of an existing plan relative to the event to be explained. Whenever there is such an active plan, each new utterance to be explained will relate to that plan in one of these three ways. As there is usually one or more active plans at any point in a dialogue, this aspect of grounding is widespread. The following sections describe each of these types of grounding in detail.

3.1.1. *Plan continuation*

A good explanation of an event includes every plan in which the event is a step. If such a plan is part of the system's existing model of the user's plans and goals, then the explanation is thereby grounded in prior knowledge. For example, consider the exchange:

UC: Is the file in your directory?

User: Yes.

UC has initiated the plan of first asking a question then receiving a response, so as to achieve the goal of knowing whether the file under discussion is in the user's directory. Prior to the processing of the user's utterance, the second step of this plan (a yes/no response by the user) has not yet been observed. Therefore, UC's question has set up a specific expectation about how the user will proceed, namely that she will provide a yes/no answer. A good explanation of the user's utterance in this example will include UC's plan, because the uttering of an affirmative response is a continuation of that plan.

It is important to note that this type of grounding is applicable to any goal that has been inferred to help explain an utterance, not just the lowest-level one. For instance, if the user had instead responded to the above question:

UC: Is the file in your directory?

User: Is the Pope Catholic?

the expectation of a yes/no response could not be used directly to ground the explanation that the speaker wants to ask a rhetorical question about the Pope's religion. However, asking such a question is a plan for the goal of conveying an affirmative response. This goal can be grounded in the expectation of a yes/no response by plan continuation. The explanation that the speaker wants to ask a rhetorical question is therefore *indirectly* grounded in the expectation. Thus, a good explanation of the user's utterance in this example would indicate that the question was asked as a plan for the goal of conveying an affirmative answer, and that conveying an affirmative answer was done as a plan for the goal of UC knowing the answer to its question.

In dialogue, plan continuation may be signaled by a participant before it occurs:

UC: What is the name of the file?

User: Hold on, let me check.

Here, the user has initiated a plan of informing UC about an upcoming plan continuation. The remaining steps of the user's plan are to obtain the requested information, then to inform UC. This plan as a whole constitutes a continuation of UC's plan.

3.1.2. *Plan delay*

When one participant in a dialogue (or in fact in any cooperative venture) initiates a plan that requires the participation of other agents, it does not follow that the other participants will readily accept the plan and agree to continue it. They may instead decide that for some reason they should take another action. There are two general categories into which phenomena of this type may be placed: plan delay, and plan rejection. Plan delay, an interruption in the execution of a plan, is the subject of this section. Plan rejection, a termination of the execution of a plan by one of its participants, is discussed in the next section.

Plan delay occurs when a plan participant wishes to delay the execution of the plan until some later time. There are three types of plan delay:

1. Clarification delay
2. Skepticism delay
3. Extra-schematic delay

A clarification delay is a delay designed to fill a gap in an agent's knowledge of the plan or goal. A skepticism delay is one in which the agent fully understands the plan or goal, but questions its validity. Finally, an extra-schematic delay is one that arises due to circumstances outside of the plan being delayed. These three types of plan delay are the subject of the following sections.

Clarification delay
Clarification delay is delay designed to fill a gap in a participant's knowledge of the plan. Consider the exchange:

UC: What is the name of the file?

USER: Do you mean the file I want to copy?

In this example, the user is not sure what question is being asked. Rather than selecting a particular interpretation of the question and answering it, thereby continuing the plan, the user decides to get clarification about the meaning of the question from UC. This delays UC's plan until the clarification is made.

Clarification delay is terminated (that is, the original plan continues) when the subplan is completed via plan continuation (see above), or when it is discarded via plan rejection (see below). In this example, once UC indicates the intended meaning of the question, the user can continue UC's original plan.

Skepticism delay
Skepticism delay arises when a plan participant fully understands the plan being delayed, but is skeptical about its efficacy or efficiency:

UC: What is the name of the file?

USER: I'm not sure I have it.

Here, the user is questioning whether UC's plan will work at all. The user can also suggest that there might be a more efficient plan:

UC: What is the name of the file?

USER: Can't you look for it?

In this example, the user is delaying UC's plan by questioning whether it wouldn't be more efficient for UC to search out the information itself.

Skepticism delay is completed, and the delayed plan is continued, when the skepticism is allayed. Since it is possible that the skepticism might not be allayed, the system must have the ability to infer that plan delay has been converted to plan rejection. Plan rejection is discussed below.

Extra-schematic delay
The third kind of plan delay is extra-schematic delay. Extra-schematic delay is plan delay that arises due to concerns outside of the plan being delayed:

UC: What is the name of the file?

USER: That reminds me of the one about the file and the Stanford student...

In this example the plan delay is triggered by a component of UC's plan, but the plan introduced by the user does not itself bear on UC's plan.

3.1.3. *Plan rejection*

The other way that an utterance may fail to continue an existing plan is by rejecting the plan as flawed in some way. As with skepticism delay, plan rejection may stem either from efficacy or efficiency considerations. A plan participant may reject a plan for reasons of efficacy either because the plan cannot be carried out, or because one of the effects of the plan is undesired. For example, a participant might be unable to answer a question:

UC: What is the name of the file?

USER: I don't know.

Here, UC has asked the user for the name of some file under discussion. UC now expects the user to tell it the name. Since the user doesn't know what the name is, she rejects the plan by indicating her inability to provide the answer.

A plan participant may reject a plan for reasons of efficiency either because there is a better way to achieve the desired outcome, or because the execution of the selected plan would be too expensive in an absolute sense. For example, the participant might indicate that another agent would be better able to perform a portion of the plan:

UC: What is the name of the file?

User: You should ask Marcia.

Here, a new plan is suggested by the user, thereby rejecting UC's initial plan.

3.1.4. *The relationship between plan delay and plan rejection*

Although it is useful to treat plan delay and plan rejection as completely separate phenomena, they are in fact closely related. This is because there is never a guarantee that a delayed plan will be resumed. Thus, when the system detects plan delay, it must have the ability to treat it as plan rejection. Furthermore, plan rejection will never be absolute, because the speaker always has the option of taking it up again. Thus the distinction between plan delay and plan rejection is one of degree.

3.2. Connecting an explanation to a goal

There are three ways that an event may attach to a goal. These ways parallel the ways that an event can attach to a plan:

1. Goal achievement
2. Goal scrutiny
3. Goal rejection

These three types of attachment to a goal are described in the following sections.

3.2.1. *Goal achievement*

An action can achieve a goal, either by continuing an existing plan, or by introducing a new plan for an existing goal (*cf.* Wilensky (1983)). The former case occurs when an action is grounded by plan continuation. The latter case occurs when an action does not conform to a specific expectation, and instead addresses a higher-level goal (that is, a goal that motivated the plan containing the expected action). Suppose that instead of responding "yes," as in an earlier example, the user engages in the following dialogue:

UC: Is the file in your directory?

User: It's in /tmp.

Many accounts of yes/no questions have tried to view a response such as this one as an answer to the yes/no question asked. Such approaches require a refinement of the notion of what it means to be a yes/no question. For example, Hirschberg (1984) represents yes/no questions as scalar queries rather than as questions that take simple yes/no answers.

In contrast to such positions, I believe it is important to view yes/no questions themselves as questions that can only take yes/no answers or answers that can be inferred to indicate yes/no answers *independent of the context of the question* (as in the response 'Is the Pope Catholic?'). The real complexity inherent in yes/no questions arises because yes/no questions are typically used in service of higher-level goals. A *response* to a yes/no question can address such a higher-level goal, but an *answer* to a yes/no question can only be yes or no.

The above example is a case in point. There is no way to construe this response as either an affirmative answer or a negative answer to the question.[2] Thus, the lowest-level expectation is not met. However, the goal UC was addressing in asking the question was to know the location of the file. This goal is itself an expectation, because it is a step of a higher-level plan that has not yet been completed. By informing UC of the file's location, the response succeeds in addressing this goal, independent of UC's original plan of receiving a yes/no response to its question; the utterance therefore matches the expectation that the user will address the goal. Thus, a response such as this one is grounded both by goal achievement, and by plan rejection.

In general, it is not acceptable simply to respond negatively to a question such as this one:

UC: Is the file in your directory?

USER: No.

This is because such a question is usually in service of the goal of knowing the location of the file, and whereas an affirmative response satisfies this higher-level goal, a negative response does not. However, there are cases where a negative response is perfectly appropriate. For example, there are some UNIX utilities that require that certain files are located in the user's directory. If the user were inquiring as to why such a utility wasn't working, then the above exchange is perfectly reasonable. The point here is that it is not a plan recognizer's responsibility to determine whether a response such as this one is malformed. Rather, this task should fall to a separate component that detects user misconceptions [*cf.* discussion in Section 2].

[2] In most cases, this response *implies* a negative answer, but only by virtue of the analysis described here.

3.2.2. *Goal scrutiny*

The second way that an event can be grounded in a goal is by questioning that goal in some way. Consider for example the exchange:

> UC: What is the name of the file?
>
> USER: Why do you want to know?

Here, the user is skeptical of UC's need for the information it is requesting, and so questions whether UC's goal is a legitimate one. Notice that in addition to questioning UC's goal, this response also delays UC's plan.

3.2.3. *Goal rejection*

The third way that an event can be grounded in a goal is by rejecting that goal. This may occur either because the goal should not be achieved, or because it cannot be achieved. For example, in the following exchange the user indicates that there is a reason that UC's goal should not be fulfilled:

> UC: What is the name of the file?
>
> USER: Sharon doesn't want anyone else to know.

Thus, UC's goal is rejected.

The other reason that a participant may reject a goal is if the participant believes that the goal cannot be achieved. For example, the goal might not be achievable if some premise of the goal is incorrect. Consider the exchange:

> UC: What is the name of the file?
>
> USER: I haven't typed the data into a file yet.

Here, UC has been led to believe that the data are contained in a UNIX file. This premise is incorrect, and consequently it is impossible for the user to answer the question. Instead, the user indicates the reason that the question is flawed. The extreme case of this phenomenon is when the user's overall goal has been incorrectly analyzed:

> UC: What's the name of the file you want to delete?
>
> USER: I don't want to delete anything.

Here, UC has incorrectly inferred that the user desires to delete a file, and has initiated a plan to determine which file is to be deleted. Since the user never intended that a file be deleted, she rejects UC's goal.

4. The principle of completeness

The principle of *completeness* states that a good explanation of an utterance covers every aspect of the utterance; it leaves no portion of the utterance, or of the explanation itself, unexplained. There are two kinds of completeness:

1. Depth Completeness
2. Breadth Completeness

Depth completeness is completeness of a single line of explanation of an utterance. An explanation of an utterance exhibiting depth completeness includes a goal that motivated the production of the utterance, and a goal or theme to explain each inferred goal. Breadth completeness is coverage of all aspects of the utterance. An explanation that exhibits breadth completeness includes every goal that motivated the production of the utterance. The following sections describe these two kinds of completeness.

4.1. Depth completeness

When an explanation of an utterance is inferred, that explanation itself may be subject to explanation. For example, the goal of requesting that UC tell the user how to compress a file might explain the utterance:

> USER: Can you tell me how to compress a file?

That goal itself can be explained by the user's goal of knowing how to compress a file, which might in turn be explained by the user's goal of compressing a file, *etc.* The depth completeness criterion states that a good explanation includes an explanation of every goal inferred in this way.

Of course, if the only type of explanation for a goal were another goal, strict adherence to the depth completeness criterion would require infinite explanations. There are two ways around this apparent quandary. First, a non-intentional explanation (Schank and Abelson, 1977; Wilensky, 1983) can be used to explain a goal, thereby terminating the chain of explanation. Secondly, the principle of applicability can be called into play. The principle of completeness is often at odds with the principle of applicability. Where the applicability criterion dictates that a particular portion of an explanation should be omitted, the completeness criterion dictates that it should remain a part of the explanation. In such cases, the applicability criterion takes precedence;

there is little sense in having a complete explanation of an utterance if it isn't applicable to the task at hand. The *modified* depth completeness criterion is then that a good explanation includes an explanation of every inferred goal *in the domain of discourse.*

4.2. Breadth completeness

The breadth completeness criterion requires that a good explanation of an utterance or of an inferred goal include every motivating goal of that utterance or inferred goal. It is derivative of Wilensky's notion of exhaustion (Wilensky, 1983). The breadth completeness criterion is based on a general principle of planning, namely that it is often possible to address more than one goal with a single action. Appelt (1985) and Norvig (1988) describe this phenomenon. For example, a person can both ask what vi does, and indicate a suspicion that it is an editor, with a question like:

> User: Is vi an editor?

The most obvious way to address two goals with a single utterance is by explicitly listing two requests, as in:

> User: I want to know how to rename a file and how to copy a file.

One might claim that such a statement is no more than a weak association of two separate statements within the same sentence. However, until a principled way is available to determine how a given input should be divided before it is processed, sentences will remain the natural processing unit. Thus, an utterance such as this one must be handled as a single request, since it uses a single sentence. A complete explanation of this statement must then include both the goal of knowing how to rename a file, and the goal of knowing how to copy a file.

The incorporation of multiple goals into a single utterance need not be so straightforward. For example, a single utterance may be used both to request something of the hearer, and to inform the hearer of a particular fact. The question:

> User: How do I delete the file 'binkle' in my top-level directory?

both requests that the hearer say how to delete the file, and informs the hearer of the whereabouts of the file. A good explanation of this question must include both of these as goals of the speaker.

It is possible to use a single utterance to address virtually any pair of goals that can themselves be addressed linguistically. Thus a taxonomy of pairs of goals that might be addressed by a single utterance is no easier to come by than a taxonomy of goals themselves. Consequently, I will not expound on such a taxonomy beyond providing a number of examples. The interested reader is referred to Appelt (1985) for a description of this problem from a generation perspective.

To see the importance of recognizing multiple goals, consider this example:

> USER: I want to delete a directory and all the files in it.

The user here has expressed two goals explicitly in a single utterance. To successfully respond to this utterance, the system must recognize *both* goals. If the system recognizes only the goal of deleting the directory, it may come up with the plan of moving its files elsewhere and then deleting it; if it recognizes only the goal of deleting the files, the plan it chooses will almost certainly leave the directory intact. Thus, it is important for PAGAN to infer *all* the goals that motivate the production of an utterance.

Consider another example:

> USER: How can I prevent anyone from reading my file without deleting it?

The user has two goals in making this utterance: to prevent other people from reading the file, and to preserve the file. Once again, the goals are distinct and non-conflicting, and both must be addressed to adequately respond to the question. The first goal is the user's main goal; the second is called an *adjunct goal* (Wilensky, 1983). However, the method for inferring these goals, given a representation of the utterance, is less straightforward than in the previous example, since they are not mentioned as a statement of goals but rather as a question.

Addressing multiple goals with a single action is not limited to goals in the domain of discourse. Discourse goals themselves can be pursued in this manner:

> USER: My last question is how to save my file once I've edited it.

The speaker of this utterance has the domain goal of finding out about how to save a file while in the editor.[3] She also has the discourse goal of indicating that she expects the dialogue to terminate after this problem has been resolved. While awareness of this latter goal may not be useful

[3] Of course, the speaker also has the discourse goal of asking how to save the file. However, this goal is subservient to the domain goal. I am concerned in this section only with multiple goals that are not causally related to one another.

to a system that simply performs rote question answering, it will be helpful to a more advanced system in planning its own utterances.

Finally, consider the statement:

USER: I have a problem with ls.

The user who initiates a conversation with UC in this way probably has several goals in mind, including to initiate a dialogue with UC, to solicit help from UC, and to inform UC that the 'ls' command is the topic of the problem. In many systems, goals such as these are not explicitly inferred. Rather, the system embodies assumptions that its authors have made about the types of higher-level goals that the system's users will have. Encoding this type of knowledge procedurally is not necessarily bad. However, the system should have the ability to represent and treat this knowledge declaratively should the need to do so arise.

The validity of the breadth completeness criterion is based on the assumption that the interests of the system that will use the explanation are not so constrained as to be single-minded. If the system *is* single-minded about its goals, then this criterion isn't useful. For example, a simplistic database lookup program will generally only need to extract a database query from the user's utterance. It won't matter to such a program whether a full understanding of an utterance is achieved, only that a database query is selected. For systems with broader applicability though, this is an important criterion.

5. System evaluation: Explanations in UC

As an example of the use of these criteria in system evaluation, in this section I will apply them to the output of UC's plan recognition component PAGAN. Two factors allow PAGAN to build explanations that meet the criteria discussed above:

1. PAGAN's representation of plans and goals

2. PAGAN's algorithm

These factors are the subjects of the next two sections.

5.1. Representation of plans and goals

There have been two main overlapping approaches to the representation of knowledge about plans and goals. Operators (as used in

STRIPS (Fikes and Nilsson, 1971) and subsequent systems, notably that of Allen and Perrault (1980)) capture knowledge about the effects of actions. They are useful for planning, but are less useful in plan recognition because they do not highlight the outcome the planner typically aims to achieve in using the operator. *Scripts* are an alternative representation. Scripts (Schank and Abelson, 1977) allow knowledge of the actions of multiple agents to be collected into a single representation. They are useful in understanding stereotypical situations, but are less useful for handling novel situations.

The structure that PAGAN uses to represent knowledge about plans and goals is called a *planfor* (*cf.* Wilensky (1983)). A planfor is a relation between a type of goal and a hypothetical event (called a plan) that constitutes a possible method of achieving a goal of that type. Typically, such plans represent compound events composed of a partially ordered set of hypothetical actions. Planfors unite the idea of a script as a way to group related actions of multiple agents with the notion of a plan that is being pursued by a single actor in order to achieve a particular goal. Planfors highlight the notion that a plan is a way to achieve a goal, as opposed to an abstract structure bearing no direct explicit relation to a goal. They also allow that a planner may intend that another agent perform some action as a part of the plan, without being in a position to control the other agent's performance of the action. Planfors are described in detail in Mayfield (1989).

The use of planfors allows PAGAN to build explanations that meet the applicability of composition criterion. UC's task is in part to address the user's goals. By emphasizing the importance of motivating goals in plan recognition, planfors give UC exactly the information it needs to provide the user with intelligent responses. In another type of system, the relationship between a plan and its motivating goal might be less important. For example, Quilici *et al.* (1985) are interested in producing explanations where causal relationships are as important as intentional ones. Their system therefore has a wide variety of link types that are used in the construction of an explanation.

Applicability of granularity is facilitated by the choice of an appropriate set of planfors for inclusion in a particular system. In PAGAN, as in most plan recognition systems, planning knowledge is hand-coded. This leads to explanations that better meet the applicability of granularity criterion, but at the expense of producing a large body of planning knowledge in an ad hoc manner. For large systems, the process of creating planfors will need to be automated. Results in explanation-based learning (see for example Braverman and Russell (1988)) show promise in this regard.

5.2. PAGAN's algorithm

Plan recognition has received much attention in recent years. Wilensky's PAM system (Wilensky, 1978) demonstrated that a simple recognition algorithm can perform sophisticated analysis of action given the appropriate knowledge about plans and goals. Allen (1979) showed how the plan recognition paradigm can be applied to language understanding. More recent work has expanded on these basic ideas (for example, Appelt and Pollack (1992), Carberry (1986), Eller and Carberry (1992), Grosz and Sidner (1985), Litman (1985), and Raskutti and Zukerman (1991)).

PAGAN's brand of plan recognition is called *goal analysis*, to stress the importance of the user's goals to the overall plan structure. Before goal analysis can begin, the user's utterance must be read and converted (via parsing and semantic interpretation) to an internal representation. This process produces one or more interpretations of the user's utterance. These interpretations are marked as being mutually incompatible before being handed to PAGAN for goal analysis. This allows PAGAN to address ambiguities that prior components of the system have been unable to resolve.

The first step of the goal analysis algorithm is to find one or more potential explanations of the utterance. There are three places where PAGAN might find an explanation for an utterance:

1. Among its expectations of speaker actions
2. Among the abstract planfors in its knowledge base
3. Among the non-intentional explanations in its knowledge base

First, an explanation might be found among the expectations that are held about future actions of the speaker. Such expectation-matching builds explanations that meet the grounding criterion, because they relate the utterance to the previous dialogue structure. For example, if the system were expecting an answer to a yes/no question, the statement:

User: No.

could easily be matched against that expectation.

Secondly, an explanation might be found among the collection of planfors stored in the long-term knowledge base. The importance of the planfor representation to the production of explanations that meet the applicability of composition and applicability of granularity criteria was described in Section 5.1.

Finally, an explanation might be found among the set of non-intentional explanations in the knowledge base. For example, the non-intentional explanation 'computer users often want to edit files' might be used to explain why a particular user wants to edit a particular file. The use of a non-intentional explanation does not necessarily assist in building explanations that meet the criteria discussed above, but it does offer a method for terminating a sequence of inferences about an utterance.

The second main step of the goal analysis algorithm is to handle any ambiguities that have arisen. Ambiguity can arise because the sentence used in the utterance is ambiguous, because the use of the sentence is ambiguous, or because the utterance as a whole (or one the goals inferred to explain the utterance) might be part of more than one plan. The sources and resolution of ambiguity are described fully in Mayfield (1989).

Sometimes, an utterance may legitimately have more than one explanation. When PAGAN produces two explanations that are compatible with one another, both are carried forward. The ability to produce multiple explanations for a single event allows PAGAN to meet the breadth completeness criterion.

The third step of the goal analysis algorithm is to determine whether the inferred explanation for the utterance should itself be explained. This decision is based on an assessment of the depth completeness of the explanation chain. An explanation that is complete is not extended any further. An explanation is complete when it meets some expectation instantiated by previous dialogue, or when the most recently inferred goal lies beyond the system's domain of expertise. An explanation that is incomplete is subjected to further analysis by the algorithm. An explanation is incomplete when the inferred goal is always used in service of some other goal, or when the dialogue model contains a strong expectation that has not yet been met. Thus, the third step of the algorithm helps to build explanations that meet the applicability of content and depth completeness criteria.

If neither of these conditions holds, then it is not immediately clear whether the explanation is complete. In such cases, two additional assessments are made to determine whether processing should continue. First, the system determines its level of curiosity about the explanation. This is compared against an estimate of the difficulty of continuing the analysis. If the level of difficulty is higher, processing is terminated. In such cases, the production of a good explanation is sacrificed in order to conserve system resources. On the other hand, if the level of curiosity about the explanation is higher, processing continues, and an explanation that better meets the depth completeness criterion is produced. See Mayfield (1992) for a complete discussion of PAGAN's chaining algorithm.

6. Computer example

The following trace shows PAGAN's processing of the user's utterance in the exchange:

UC: Is the file in your directory?

User: The file is in /tmp.

The trace shows how PAGAN is able to produce explanations that meet the applicability, grounding, and depth completeness criteria. While the explanation produced by PAGAN meets the breadth completeness criterion, it does so only because there is just one explanation chain to be derived. An utterance that addressed more than one compatible goal would have led PAGAN to build multiple compatible explanations, better illustrating compliance with the breadth completeness criterion.

The following conventions are used in the traces: concept nodes in KODIAK (the semantic net representation language used in UC (Wilensky, 1987)) are written in capital letters. Descriptions produced by PAGAN as it runs are written in roman, and comments added by hand after-the-fact are written in italics. Names that end with a number represent subconcepts of the concept represented by the root name. For example, LOCATION–OF8 is a specialization of the LOCATION–OF concept. Other than deletion of extraneous trace information after-the-fact, and the introduction of comments in italics, the only changes made to PAGAN's trace output by hand are changes in appearance, such as the introduction of font information and the addition of white space.

UC: Is the file in your directory?

USER: The file is in /tmp.

First, the parser produces an explanation of the user's utterance, rooted at TELL–ACTION3.

Interpretation produced by the parser: TELL–ACTION3

PAGAN *begins its processing by determining whether it really has any work to do.*

Determining whether to chain on TELL–ACTION3.
    Determining whether TELL–ACTION3 is a complete explanation of
      (THE FILE IS IN /TMP).
        Determining whether TELL–ACTION3 is outside of UC's
          domain of expertise.
        TELL–ACTION3 is not outside of UC's domain of expertise.
    The explanation is not necessarily complete.
    Determining whether TELL–ACTION3 is an incomplete explanation of
      (THE FILE IS IN /TMP).
        Determining whether TELL–ACTION3 is an instrumental goal.
        TELL–ACTION3 is an instrumental goal.
    The explanation is incomplete.
Chaining should be done on TELL–ACTION3.

PAGAN *has decided that it should try to explain the user's utterance, so it searches for an initial explanation.*

Attempting to find an explanation for TELL–ACTION3.
    Determining whether TELL–ACTION3 is grounded in the preceding
      dialogue.

Here, PAGAN *tries to determine whether* TELL–ACTION3 *is a continuation of* UC*'s plan, which calls for a yes/no answer. It does so by asserting that it is, then catching the error that results.*

    Trying to ground TELL–ACTION3 in INDICATE–YES–OR–NO.
        Concreting TELL–ACTION3 to be INDICATE–YES–OR–NO.
    TELL–ACTION3 was not grounded in INDICATE–YES–OR–NO
      because:
      Can't concrete UTTERANCE3 to UTTERANCE–OF–INDICATE–
        YES–OR–NO because the ranges differ.

Next, PAGAN *checks whether* TELL–ACTION3 *constitutes a new plan for* UC*'s existing goal of knowing whether the file is in the user's directory. It does this in the same way that it checked for plan continuation above.*

    Determining whether TELL–ACTION3 is a new plan for the existing
      goal of KNOWING–WHETHER1
    TELL–ACTION3 is not a new plan for the existing goal of
      KNOWING–WHETHER1 because:
      Attempt to concrete incompatibly from TELL–ACTION3 to
        KNOWING–WHETHER1.
    TELL–ACTION3 was not grounded in the preceding dialogue.

Now, PAGAN *looks for a planfor explanation of the utterance. It finds one, and infers that the user has the goal of* UC *knowing whether the file is in the directory. The use of this planfor leads to an explanation that meets the applicability of composition criterion (because* UC *is concerned with plans and goals) and the applicability of completeness criterion (because the planfor was created to reflect the level of granularity expected by* UC*).*

Trying to find planfor explanations for TELL–ACTION3.
Found PLANFOR53 as an explanation for TELL–ACTION3.
Synopsis:
Goal: hearer know whether X;
Step 1: Tell hearer that X.
Inferred goal is KNOWING–WHETHER2.

Explanation found for TELL–ACTION3.

PAGAN *now determines whether it should continue its processing.*

Determining whether to chain on KNOWING–WHETHER2.
Determining whether KNOWING–WHETHER2 is a complete explanation of TELL–ACTION3.
Determining whether KNOWING–WHETHER2 is outside of UC's domain of expertise.
KNOWING–WHETHER2 is not outside of UC's domain of expertise.
The explanation is not necessarily complete.
Determining whether KNOWING–WHETHER2 is an incomplete explanation of TELL–ACTION3.
Determining whether KNOWING–WHETHER2 is an instrumental goal.
KNOWING–WHETHER2 is an instrumental goal.
The explanation is incomplete.
Chaining should be done on KNOWING–WHETHER2.

PAGAN *has decided that although the explanation as it stands meets the applicability of content criterion, it does not meet the depth completeness criterion;* PAGAN *therefore continues its processing.*

Attempting to find an explanation for KNOWING–WHETHER2.
Determining whether KNOWING–WHETHER2 is grounded in the preceding dialogue.

Once again, PAGAN *checks for plan continuation.* KNOWING–WHETHER2 *is not a type of indicating yes or no, so the check fails.*

Trying to ground KNOWING–WHETHER2 in INDICATE–YES–OR–NO.
KNOWING–WHETHER2 was not grounded in INDICATE–YES–OR–NO because:
Attempt to concrete incompatibly from KNOWING–WHETHER2 to INDICATE–YES–OR–NO.

Now, PAGAN *checks whether* KNOWING–WHETHER2 *represents a new plan for* UC*'s existing goal. It does, so the explanation is grounded in the representation of the dialogue so far. The old plan is rejected, and processing terminates.*

Determining whether KNOWING–WHETHER2 is a new plan for the existing goal of KNOWING–WHETHER1
KNOWING–WHETHER2 is a new plan for the existing goal of KNOWING–WHETHER1
Existing plan PLAN107 has been rejected; new plan is PLAN111.
Explanation found for KNOWING–WHETHER2.

---

7. Conclusions

This paper introduces the following criteria for good explanations of utterances:

1. Principle of Applicability
   - Applicability of Composition
   - Applicability of Content
   - Applicability of Granularity
2. Principle of Grounding
3. Principle of Completeness
   - Depth Completeness
   - Breadth Completeness

The principle of *applicability*, states that a good explanation of an utterance is one that is applicable to the needs of the system that will use that explanation. The second criterion, the principle of *grounding*, states that a good explanation of an utterance relates what is inferred from that utterance to what is already known about the speaker and the dialogue. The last criterion, the principle of *completeness*, states that a good explanation of an utterance covers every aspect of the utterance. Together, these three criteria provide a means to determine the quality of an explanation of an utterance. They are useful because they can be used to evaluate the effectiveness of a plan recognition algorithm at a high level, and to suggest how such an algorithm might be improved.

References

James F. Allen. *A Plan-Based Approach to Speech Act Recognition.* Ph.D. thesis, University of Toronto, Computer Science Department, 1979. Technical Report 131/79.

James F. Allen and C. Raymond Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178, 1980.

Douglas E. Appelt. Planning English referring expressions. *Artificial Intelligence*, 26(1):1–33, 1985.

Douglas E. Appelt and Martha E. Pollack. Weighted abduction for plan ascription. *User Modeling and User-Adapted Interaction*, 2(1–2):1–25, 1992.

Michael S. Braverman and Stuart J. Russell. IMEX: Overcoming intractability in explanation based learning. In *Proceedings of the National Conference on Artificial Intelligence*, Los Altos, CA, 1988. Morgan Kaufmann.

Randall J. Calistri-Yeh. Utilizing user models to handle ambiguity and misconceptions in robust plan recognition. *User Modeling and User-Adapted Interaction*, 1(4):289–322, 1991.

Sandra Carberry. TRACK: Toward a robust natural language interface. In *Proceedings of the Canadian National Conference on Artificial Intelligence*, 1986.

David Ngi Chin. *Intelligent Agents as a Basis for Natural Language Interfaces.* PhD thesis, University of California, Berkeley, Computer Science Department, 1988. Technical Report UCB/CSD 88/396.

Rhonda Eller and Sandra Carberry. A meta-rule approach to flexible plan recognition in dialogue. *User Modeling and User-Adapted Interaction*, 2(1–2):27–53, 1992.

Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

Barbara Grosz and Candace L. Sidner. The structures of discourse structure. Technical Report CSLI-85-39, Center for the Study of Language and Information, 1985.

Julia Hirschberg. Toward a redefinition of yes/no questions. In *Proceedings of the Tenth International Conference on Computational Linguistics*, 1984.

Robert Kass and Tim Finin. Modeling the user in natural language systems. *Computational Linguistics*, 14(3):5–22, 1988.

Diane J. Litman. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues.* PhD thesis, University of Rochester, 1985. Technical Report TR170.

James Mayfield. *Goal Analysis: Plan Recognition in Dialogue Systems.* PhD thesis, University of California at Berkeley, Computer Science Division (EECS), 1989. Technical Report UCB 89/521.

James Mayfield. Controlling inference in plan recognition. *User Modeling and User-Adapted Interaction*, 2(1–2):83–115, 1992.

Peter Norvig. Multiple simultaneous interpretations of ambiguous sentences. In *Program of the Tenth Annual Conference of the Cognitive Science Society*, 1988.

Martha E. Pollack. Good answers to bad questions: Goal inference in expert advice-giving. Technical Report MS-CIS-84-15, University of Pennsylvania, Computer Science Department, 1984.

Alexander E. Quilici. Detecting and responding to plan-oriented misconceptions. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems*. Springer Verlag, 1989.

Alexander E. Quilici, Michael G. Dyer, and Margot Flowers. Understanding and advice giving in AQUA. Technical Report UCLA-AI-85-19, University of California, Los Angeles, Computer Science Department, 1985.

Bhavani Raskutti and Ingrid Zukerman. Generation and selection of likely interpretation during plan recognition in task-oriented consultation systems. *User Modeling and User-Adapted Interaction*, 1(4):323–353, 1991.

Gudula Retz-Schmidt. Recognizing intentions, interactions, and causes of plan failures. *User Modeling and User-Adapted Interaction*, 1(2):173–202, 1991.

Roger Schank and Robert Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum, Hillsdale, NJ, 1977.

Roger C. Schank. *Explanation Patterns: Understanding Mechanically and Creatively*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

Robert Wilensky. *Understanding Goal-Based Stories*. PhD thesis, Yale University, Computer Science Department, 1978. Research Report 140.

Robert Wilensky. *Planning and Understanding: A Computational Approach to Human Reasoning*. Addison-Wesley, Reading, MA, 1983.

Robert Wilensky. Some problems and proposals for knowledge representation. Memorandum UCB/CSD 87/351, University of California, Berkeley Electronic Research Laboratory, 1987.

Robert Wilensky, Yigal Arens, and David Chin. Talking to UNIX in English: An overview of UC. *Communications of the ACM*, 27(6):575–593, 1984.

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

Address for correspondence: James Mayfield / The Johns Hopkins University Applied Physics Laboratory / Laurel, MD 20723–6099 USA / James.Mayfield@jhuapl.edu