Preprint of final version appearing in Artificial Intelligence, no. 115, 215-256, 1999

Using Grice's Maxim of Quantity to Select the Content of Plan Descriptions

R. Michael Young

Department of Computer Science, North Carolina State University, Raleigh, NC, 27695-7534

Abstract

Intelligent systems are often called upon to form plans that direct their own or other agents' activities. For these systems, the ability to describe plans to people in natural ways is an essential aspect of their interface. In this paper, I present the Cooperative Plan Identification (CPI) architecture, a computational model that generates concise, effective textual descriptions of plans. In this model, speakers and hearers cooperate with one another in their communication about a plan. A hearer interprets a concise plan description by filling in the missing detail using plan reasoning. A cooperative speaker selects the content of a plan description based on his expectation that the hearer is able to complete the description in much the same way that a planning system completes a partial plan. The architecture has been empirically evaluated in an experiment, also described here, in which subjects following instructions produced by the CPI architecture performed their tasks with fewer execution errors and achieved a higher percentage of their tasks' goals than did subjects following instructions produced by alternative methods.

Key words: Natural Language Processing; Task-Related Discourse; Planning; Plan-Space Search.

1 Introduction

Complex activities, by definition, contain a large amount of detail. When people describe activities to one another they leave out information they feel is unimportant and emphasize information they feel is essential. This economy of communication is an example of speakers obeying Grice's maxim of Quantity: speakers should say no more and no less than what is needed [16]. There is a wide range of contexts where computers that create and use plans might

require the ability to generate task descriptions of similar brevity. Unfortunately, it is not a straightforward matter to produce an effective description of a given plan automatically when one or more of the intended readers or hearers are human. There is a mismatch between the amount of detail in a plan for even a simple task and the amount of detail in typical plan descriptions used and understood by people.

The quality of a textual description is strained when it contains too little information. For instance, providing too little detail may so underconstrain the interpretation process that the hearer's plan reasoning resources are overburdened. Further, too little detail may result in the hearer inferring a plan that is incompatible with the one that the speaker intends. On the other hand, a description's effectiveness is also strained when it contains too much detail. Providing a hearer with too much detail may needlessly cause her to eliminate from consideration compatible alternate plans or may give her so much information that her attentional constraints are overtaxed.

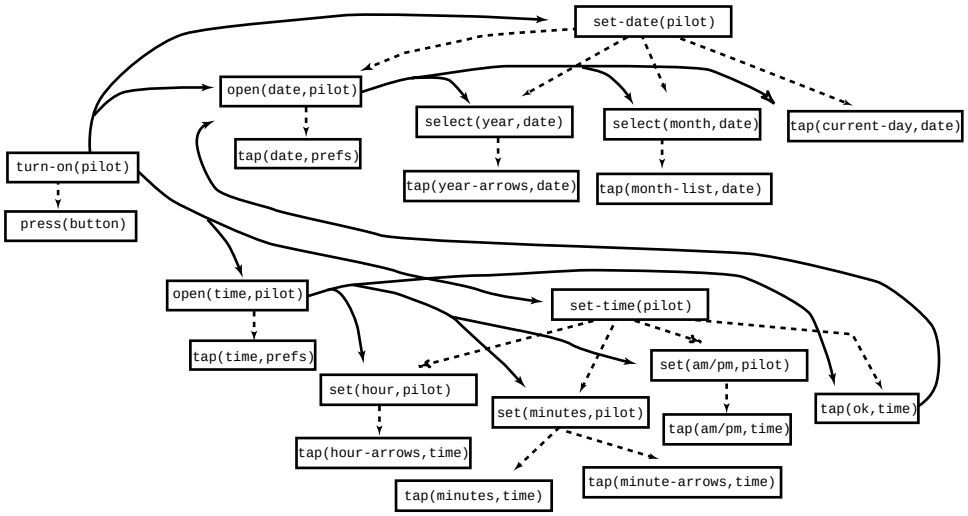

Fig. 1. A Sample Plan: Configuring a PalmPilot Organizer.

Figure 1 provides a graphical representation of the data structures that a machine might use to represent a simple task: initializing a PalmPilot Orga-

nizer.[1] This plan contains several steps that involve turning the Pilot's power on and setting the Pilot's internal time and date. In addition to the steps in the plan, the data structure contains a number of constraints on those steps. The plan contains temporal constraints over the steps to ensure that they are executed in the correct order. It contains variable bindings to associate the variables in each of the steps with the proper objects in the domain. Because the plan is hierarchically structured, the plan contains links that relate an abstract step to the steps that play a role in its immediate subplan. Finally, it contains links used to indicate dependencies between one step's effects and the preconditions of subsequent steps.

All told, the data structure for the plan contains considerable detail, detail that is added by the planning system that produced the plan for a very specific reason. Without the steps and other components of the plan, the planning system cannot ensure that the plan is sound, that is, that its steps will successfully execute and, when the plan terminates, that the goals of the problem will all be satisfied. A textual description of this plan that describes *all* of its components appears in Description 1.

Description 1: First, turn on your PalmPilot in order to bring up the General Preferences screen. To turn on your PalmPilot, press the green power button located on the front of the Pilot. Next, set the current time by first tapping the time shown in the General Preferences Screen. This will open the Set Time dialog. Then set the hour, the minutes and finally the time of day (AM/PM). To set the hour, tap the up or down arrows next to the hours setting. To set the minutes, first tap the minutes setting, then tap the arrows next to the minutes setting. Tap the AM or PM button as appropriate to set the time of day. Tap the OK button to close the Set Time dialog and return to the General Preferences Screen. To set the

[1] The plan in this figure is represented as a DPOCL [42] data structure (DPOCL is described in more detail in Section 4.1.3). In the figure, rectangles represent steps in the plan, dashed arcs from one step to another indicate that the second step plays a role in the first step's subplan, solid arcs from one step to another indicate that the first step establishes a condition needed by the second step and left-to-right spatial ordering roughly indicates the temporal order of execution of the steps.

current date, first tap the date. This will open the Set Date dialog. Then select the current year by tapping the arrows at the top of the screen. Next, tap the listing of months to select the current month. Finally, tap the current date.

It is clear that this description contains an unnatural amount of detail, particularly when we compare it to the instructions printed on the PalmPilot Quick Start Guide [35]. Under the "Getting Started" section, the Guide contains instructions that read: "Turn on your PalmPilot connected organizer. Press the green power button on the front of your PalmPilot. Set the data and time. The General Preferences screen appears. Set the time, date and other system preferences here." In this example, explicit reference is made only to a small subset of the plan components in the plan from Figure 1. In line with our intuitions about the text in this example, I argue in this paper that explicit reference to each of a plan's components is counterproductive to the successful description of the plan as a whole.

Experimental evidence indicates that the descriptions typically used by people in instructional contexts are *partial*– they do not contain sufficient information for readers or hearers to carry out the plans being described should they perform just the actions contained in the descriptions. Hull and Wright [41], for example, describe a sequence of experiments that reveal that subjects give incomplete instructions in face-to-face situations as often as 58% of the time. Donin, *et al*, [10] describe results where, on average, subjects writing instructional texts provided less than half the information that the subjects themselves considered necessary for performing the task.

No one would argue that speakers who employ partial descriptions intend for their hearers to carry out only those actions explicitly mentioned in their utterances. Speakers that use these descriptions do so expecting the hearer to execute a complete plan. Given the gap between the partial descriptions of plans used by speakers and the complete plans they intend to convey, it is reasonable to assume speakers also intend for their hearers to *fill in the detail*, to reproduce the missing plan components prior to the point at which the

hearer acts on the instructions.

The use of partial plan descriptions has an intuitive justification: leaving out detail that can be recovered by the hearer makes for more efficient discourse. Of course, providing a sketchy plan, or no plan description at all, is a risky proposition. When given an extremely partial description, a hearer might not be able to piece together any complete plan from it or she might put together a plan that is considerably different from the plan that speaker was intending to convey. The speaker must balance the desire to minimize communication with the desire to provide enough information; he must find a plan description that says as little as is needed in order to successfully convey an appropriate plan.

While several natural language systems have been developed for the generation of textual descriptions of action [26,28,38], these systems have been limited in the effectiveness of the descriptions they produce by the complexity of the activities that they describe. Some of these systems have dealt exclusively with artificial plans of limited size. Because these systems avoid the complexity introduced by dealing with larger plans, it is not clear how well the techniques that they employ will scale. Other systems that have described plans of more realistic complexity have generated text as detailed as that of Description 1. In order for future systems to describe plans effectively, they must employ some technique for conveying information about the plans that communicates less than the plans' full structure. Without a principled method for determining what content to retain and what to remove, the effectiveness of textual plan descriptions will continue to be limited. The problem that this paper addresses is how to determine an appropriate subset of the components of a plan to communicate as the plan's description.

The particular plan descriptions which I focus on here occur in a context I call *plan identification.* In this context, a speaker has a particular plan in mind and describes the plan to the hearer in order to identify it as a solution to a mutually understood planning problem. The speaker selects the content of the plan's description in order to distinguish it from other plans that the hearer

might form as solutions to the planning problem.

The plan description process is modeled here as cooperative activity between a speaker and a hearer.[2] In order to understand a plan description, a hearer uses her knowledge about plans and planning to fill in any information that was missing from a speaker's description. To produce a plan description that is cooperative, a speaker uses his knowledge about the hearer's interpretation process to select a plan description that is brief but contains enough information to be understood. This work focuses on the speaker's portion of the collaboration – creating plan descriptions – but employs a model of the hearer's interpretation process in order to produce appropriate descriptions. In this model, a hearer's interpretation of a description involves reasoning about its underlying plan — a partial plan description poses a planning problem for the hearer and in order for her to understand the description, she must fill in the details that are missing. To complete a partial plan description, the hearer performs what is essentially the same type of planning activity that she would perform to create the plan herself.

A cooperative speaker can anticipate the hearer's interpretation of a description by anticipating the outcome of her plan reasoning process and adjusting the content of the description accordingly. Of particular importance to the speaker are two aspects of the hearer's plan reasoning process: her preferences over plans and her plan reasoning resource bounds. In a cooperative approach to plan description, these two aspects of the hearer's interpretation process influence the content of a description in the following ways. First, a description should not be so incomplete that the hearer's task of reconstructing a complete plan from it exceeds her resource bounds. Second, the content of a description should be such that the plan that the hearer settles on after filling in any missing detail is reasonably close to the plan that the speaker was describing. Finally, a description should contain no more detail than is needed to meet the first two requirements above. Taken together, these three criteria define

[2] For clarity, I will adopt the convention of referring to the speaker using the masculine gender while using the feminine gender for references to the hearer.

what I mean when I say that a description is cooperative. Speakers that take these criteria into account when producing plan descriptions perform what I term *cooperative plan identification.*

In the discussion that follows, I define a computational model of cooperative plan identification, give examples of its use and evaluate the descriptions that it produces. I also describe an empirical evaluation of the cooperative plan identification techniques where plan descriptions are used as instructions for tasks carried out by human subjects. Subjects carry out their tasks in a text-based virtual environment and their performance on the tasks is measured to determine the effectiveness of the instructions that they follow. The experiment demonstrates that the descriptions produced by cooperative approaches are more effective than those produced by several alternative techniques.

2 Related Work

The principal work on describing plans produced by AI planning systems was done by Mellish and Evans [26]. Their system produces a textual description of a plan produced by the NONLIN planner [32]. The plans used by Mellish and Evans often contained a deep hierarchical structure, introduced by NONLIN as a result of the numerous levels of abstraction present in their domain's plan operators. Aside from eliminating incidental artifacts of the planning process, their program does little to remove detail from the message at any point, resulting in plan descriptions that make reference to every component in a plan. As Mellish and Evans point out, this results in descriptions that often contain an inappropriate amount of detail.

Other researchers have investigated the generation of descriptions of artificially created plans. Vander Linden [38] describes a text generation system that produces texts that describe individual plan components, focusing on the selection of rhetorical relations that best expresses the components' procedural relationships with other actions in the same plan. The Drafter project at the University of Brighton [6,28,29] has developed a system to exploit a

plan-based representation of activities to support multilingual instruction generation. This system represents task domains in a common action language and automatically generates instructions based on the plans for a given task. Because the plans that these systems use either employ simplified plan representations or are pre-defined by human designers rather than by a planning system, the instructions they produce do not contain the type of complexity present in plans produced by automatic planning systems. As a result, the need for these systems to construct concise descriptions automatically is not a focus of the work.

A number of researchers [18,15,5] have investigated the generation of concise descriptions of abstract entities other than plan structures. Despite their different areas of application, the problems that each system addresses can be characterized collectively in terms of their input and output. Each system produces a textual description of a data structure created by some system that is external to and independent of the text generation system (e.g., a discourse generation system, a constraint satisfaction problem solver). I refer to these external systems as *source* systems, since they are the source of input for the text generation systems of interest to this discussion. Similarly, I refer to the the data structures that they produce (and that serve as input to the text generation systems) as *source* data structures. The source data structures are created by the external systems as solutions to problems in their own areas of application; the text generation systems provide post hoc textual interfaces for the source systems' output.

To determine what content these systems can elide from a description, they use rules that define the information that a partial description conveys implicitly. Application of these rules corresponds to the inferences that hearers will draw from incomplete texts. To produce a concise text, the systems use a generate-and-test approach, first creating partial texts as candidates and then using the inference rules to determine if the hearer is capable of inferring the elided content from the more concise discourse.

Each of these systems is motivated by Grice's *maxim of Quantity* [16]: [3]

- Make your contribution as informative as required (for the current purposes of the exchange).
- Do not make your contribution more informative than required.

In order to translate the maxim of Quantity into a computational model capable of producing concise texts, these researchers provide answers, either explicitly or implicitly, to four questions. First, they define the space of valid candidate descriptions their systems consider (some of these systems consider less than the complete space of possible descriptions). Second, they define a partial order over that set and use the ordering to characterize a candidate's quantity relative to other descriptions. Finally, they specify what constitutes a sufficient quantity of information for a text to contain in order to meet their system's communicative purpose (i.e., describing objects in its application domain). To obey their interpretations, the systems must produce texts that are minimal among all the descriptions that contain sufficient information.

Green and Carberry [15] describe a technique for generating and interpreting *indirect answers* in the form of discourse plans structured in conventional nucleus/satellite components [25]. In their approach, the components of the discourse corresponding to direct answers are removed from a discourse plan but the appropriate surrounding information is retained, creating a more concise, *indirect* response. Their generation model creates concise plans in a two-phase process. The first phase, content planning, builds a complete discourse plan whose goals address the user's question; direct answers appear in the plan's nuclei and appropriate supporting information in the satellites. In the second phase, called plan pruning, nuclei are pruned from the plan structure. To determine if a particular nucleus should be elided, their system first creates a new plan by pruning the nucleus, then uses an explicit model of the hearer's interpretation process to evaluate the understandability of the resulting plan.

[3] One exception is the system built by Horacek – his approach is motivated by Grice's maxim of Relevance, but it follows the general approach used by other researchers motivated by the maxim of Quantity.

As a result of this pruning, the process finds a subset of the original plan that contains a small number of communicative acts and yet remains understandable to the hearer.

For the generation component of Green and Carberry's system, the purpose of the exchange is for a hearer to recognize a complete discourse plan after observing the performance of a series of surface speech acts selected from the discourse plan's leaf nodes. The ordering that Green and Carberry use to characterize each candidate is a partial ordering that counts the number of speech acts in the candidate discourse plan. In their approach, a hearer that is presented with a partial discourse fills in the missing information by attempting to recognize the complete plan that underlies the partial one. The sufficiency criterion that their system uses is also cast in terms of the hearer's plan recognition – a candidate response contains sufficient detail just when the hearer can re-construct the systems' original discourse plan from the candidate.

Horacek [18] describes a system used to construct concise explanations of the answers to constraint satisfaction problems. His approach is to define a set of rules that approximate Grice's maxim of Relevance in an explanatory context, identifying information that is implicitly conveyed by the components that appear in an explanation. To generate a concise explanation, he first constructs the complete content of the solution to a constraint satisfaction problem. This solution is composed of inference rules, the facts used to satisfy the rules' premises and the conclusions reached by application of the rules. His approach iteratively considers subsets of these constraints as candidate explanations. To evaluate each candidate explanation, the algorithm iteratively applies the relevance-related rules to subsets of the constraints in order to determine the information the subset conveys implicitly; Horacek begins the generation process by starting with an empty set of constraints as the initial candidate and incrementally adds constraints until he reaches the first set whose contents implicitly conveys the full content of the original explanation.

For Horacek's system, the purpose of the exchange is for a hearer to infer a complete set of instantiated rules that make up the source data structure.

The ordering defined on this candidate set is a partial ordering counting the number of components contained in a candidate. The sufficiency criterion that his system uses to evaluate a candidate description is similarly related to the results of applying these rules: if all the components present in the original solution are inferred by the application of the rules, then the description satisfies the criterion.

In their characterization of the various computational models that have been used to generate referring expressions, Dale and Reiter [5] describe several interpretations of Grice's maxim of Quantity and discuss the computational properties of algorithms corresponding to each interpretation. These algorithms order candidate descriptions based on the number of attributes each description contains. A description contains a satisfactory amount of detail just when it is a *distinguishing description*, a description that contains a list of properties that serves to distinguish the object from all of its distractors (those objects that might also be referred to in the same context).

Two of their interpretations, relevant to the work described in this paper, are the *Full Brevity* interpretation and the *Local Brevity* interpretation. Under the Full Brevity interpretation, the shortest possible distinguishing description is selected. Because, in the worst case, an algorithm following this interpretation will have to consider descriptions containing every possible combination of an object's properties, this approach is intractable (as Dale and Reiter point out). Under the Local Brevity interpretation, the selection is made by starting with an initial distinguishing description (for instance, a description including all the object's properties) and removing individual properties based on preference rules until no further properties can be removed. This approach motivates one of the algorithms for selecting the content of a plan description described in Section 5.1.

3 A Cooperative Approach to Plan Description

In this research, I adopt the view that the use of plan descriptions in discourse is an instance of Gricean cooperation: Grice's maxim of Quantity guides a speaker when he is selecting the amount and type of detail to include in a plan description. In the context of plan identification, a speaker's purpose in his use of a plan description is to cause the hearer to adopt a mental representation of a plan that is reasonably close to the one that he's describing. I say "reasonably close" rather than "identical to" because the speaker may be willing to tolerate some variation between his plan and the plan that the hearer will adopt as a result of their communication. For instance, if I was to describe to you a plan for assembling a new TV stand, it might be of little concern to me which of the two flat-head screwdrivers in your toolbox you choose to use or whether or not you attach the front casters on the base before you attach the rear casters. The speaker will consider some aspects of the plan as essential for the hearer to adopt and some aspects unimportant. The model here allows for differentiation between the two cases.

In order for a candidate plan description to meet the speaker's communicative purpose, the hearer must be able to take the plan description and reconstruct a complete plan from it. Further, that complete plan must be similar enough to the plan being described that the speaker is willing to tolerate the hearer adopting the reconstructed plan in place of the plan that he is describing. Interpreting the maxim of Quantity in this domain, I consider a plan description *cooperative* when it contains no more and no less detail than is needed for the hearer to adopt a plan reasonably similar in structure to the plan that the speaker is describing.

To produce cooperative plan descriptions, I use a generate-and-test architecture called *cooperative plan identification* (CPI), shown graphically in Figure 2. The process used by the architecture is divided into two functions. The first, called the *generator* function, constructs candidate descriptions and the second function, called the *evaluator*, tests descriptions against success criteria

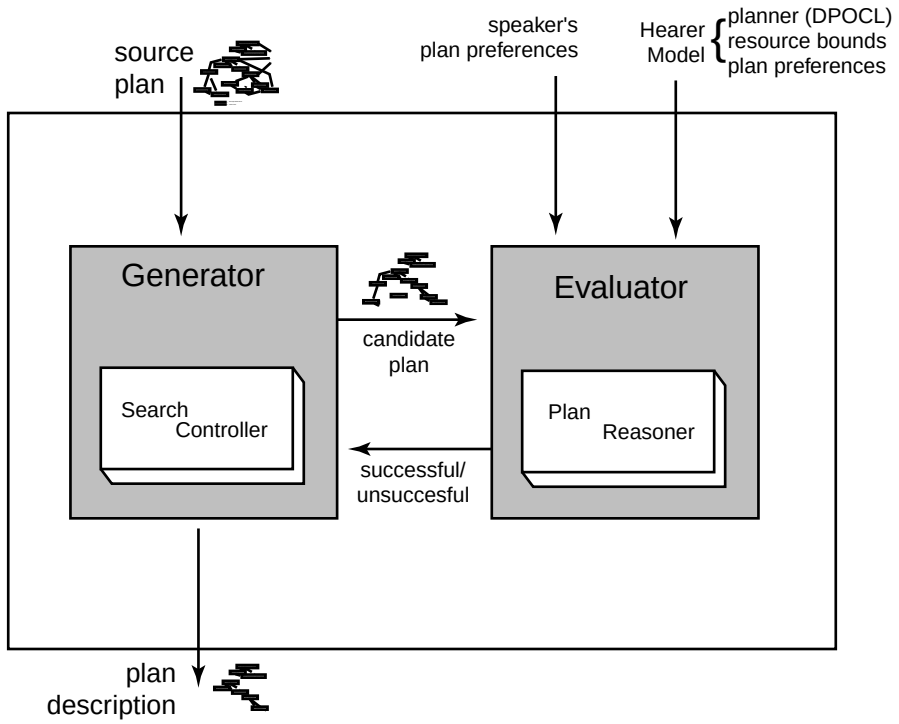

Fig. 2. An Overview of the Cooperative Plan Identification Algorithm.

captured by the interpretation of the maxim of Quantity sketched above. The algorithm's generator searches the space of descriptions of the plan that the speaker is identifying (called here the *source plan*), looking for a plan description of minimal size and structure. Searching the space of all plan descriptions is computationally expensive, since the space contains (roughly) every subset of the steps and constraints that make up the source plan. Rather than perform an exhaustive search for candidate descriptions, I describe two algorithms used as CPI generator functions that restrict the space they search to more tractable subsets of the full space while still producing reasonable candidates. [4]

The CPI evaluator function takes as input a candidate plan description and

[4] The architecture is parameterized; system developers can customize the architecture by providing application-specific components as input to a CPI implementation. One such implementation, with fully specified parameters, is used in the evaluation of this work described in Section 6.

determines if the candidate is an appropriate description of the source plan. In the CPI model, when a speaker uses a partial plan description, he poses a problem for the hearer: What is the complete plan that the speaker is describing? To determine if some description successfully identifies a particular plan, the speaker anticipates the hearer's solution to the problem by anticipating her interpretation process. This process is represented by the use of a plan reasoning algorithm that takes a partial plan description and fills in its gaps, performing the same type of plan reasoning that a planning system would use to create the source plan in the first place. If the resulting plan is similar enough in structure to the source plan, the evaluator function will characterize the candidate as acceptable.

In addition to the hearer's planning system, there are two other components that are central to the hearer's process of plan reasoning. The first is the limit on the hearer's reasoning resources — specifically, the amount of plan reasoning that she can bring to bear during the interpretation process. Clearly, the effort needed to construct a complete plan from a partial one requires the use of a hearer's reasoning resources. These resources are finite; a cooperative speaker that takes the hearer's use of these resources into account can adjust the content of his description so that the resources will not be exhausted. In particular, if a speaker suspects that the constraints in a description are so sparse that filling in the missing components will overtax the hearer's abilities, then the speaker can select an alternative description containing more detail.

The second central issue in the representation of the hearer's plan reasoning is her use of plan preferences, that is, her preferences for plans of particular structure over others. Evidence indicates that preferences between plans play a strong role in plan construction in collaborating groups [23]; the advantage of using plan preference in plan construction has also been demonstrated in the context of single-agent planning [40]. As a hearer fills in the gaps in a partial plan description, her planning activity is influenced by her preferences over aspects of the task domain. Her preferences for types of actions, for particular tools or locations, for particular sequences of actions to achieve a goal all

influence the structure of the complete plan that will emerge. By considering the hearer's preferences when composing a plan description, the speaker can prevent the hearer from forming a plan that varies in unacceptable ways from the source plan.

As an example, suppose that I am describing a plan for one of my employees to attend a business meeting in Cupertino. The plan and its description deal mostly with travel activity (taking taxis, riding on airplanes, etc). If I don't describe the airline on which my employee is to travel, she will need to select one on her own when creating a complete plan for the task. If she has a strong preference for flying US Airways (she's building up her frequent flier miles, let's say), then she will likely form a complete plan involving travel with them. If I don't care what airline she flies on, then I need not be concerned. But if I know of her plan preferences and they differ from mine, I had better provide more detail. If US Airways is the most expensive carrier and I want to keep corporate expenses to a minimum, I should add specific detail to my plan description to keep my employee from booking a flight on the more costly alternative.

The two aspects of the hearer's plan reasoning model (i.e., her resource limits and her plan preferences) do not operate in isolation from one another. Rather, they interact during her plan reasoning in ways that may seriously constrain her ability to form plans that are acceptable to the speaker. For instance, a hearer's preferences might be so strong that they guide her to search for plans that use particular actions to the exclusion of others. If no complete plans that include the preferred actions exist, she may exhaust her resources before turning to less-preferred alternatives.

In the remainder of this paper, I will refer to the source plan that the speaker is describing using the symbol $\mathcal{P}_S$, to candidate descriptions of $\mathcal{P}_S$ using the symbol $\mathcal{P}_\mathcal{C}$ and to the description that is actually used to identify $\mathcal{P}_S$ using the symbol $\mathcal{P}_\mathcal{D}$. The approach taken in cooperative plan identification is to model the communication of a plan by the communication of its constituent parts (e.g., the steps of the plan, the ordering constraints between steps, the plan's

variable bindings). Reference to an individual component in a description indicates the presence of that component in the source plan. In instructional text, for instance, utterances that describe steps in a task are often realized as imperatives like "Double-click the Find icon." Descriptions of other component types (e.g., enablement-like relations between the effects of one step and the preconditions of another) take on other forms (e.g., purpose clauses [7]). I call the types of utterances that indicate the presence of some plan component in a plan *positive constraint utterances* or simply *positive constraints*; they indicate that any plan that the hearer forms during interpretation should contain the plan components that they describe.

Descriptions of plans typically do not make reference to domain entities (e.g., objects, locations, tools) without also making reference (explicitly or otherwise) to the action(s) involving the entities. For instance, instructions for installing an internal CD-ROM into a personal computer would seem incoherent if they stated that a phillips head screwdriver and the computer's power supply were involved in the installation but never explained what role each played in the activity. Subsets of a plan that contain reference only to other members of the same subset can be thought of as forming a strongly connected graph. Descriptions of these types of plan subsets are called *referentially coherent* [21,24]. In this paper, I only consider referentially coherent descriptions of plans.

4 Computing Plan Descriptions

The cooperative plan identification architecture uses a generate-and-test approach to construct plan descriptions; the remainder of this section describes the processing performed by the two components of the CPI implementation. In order to provide a clear characterization of the types of descriptions that successfully identify a source plan, I first describe the evaluator component. Then I describe two alternative generator functions that have been implemented within the CPI architecture.

4.1 The Cooperative Principle: Characterizing a Candidate Description

Implicit in the maxim of Quantity (and, in fact, in all of Grice's maxims) is the notion that the speaker and hearer are collaborators in communication. The amount of information that is needed in an utterance is as much a function of the hearer's ability to interpret what is said as it is of the speaker's communicative goals. For the speaker to be an effective collaborator in this process, he must anticipate the hearer's interpretation when choosing the content of his plan description. In cooperative plan identification, the hearer's interpretation is anticipated by simulating her reasoning using a hearer model; the following sections describe the parts of this model and how they are used.

4.1.1 A Hearer Model Based on Plan Reasoning

The CPI architecture uses a hearer model comprised of three components that characterize the hearer's plan reasoning abilities: a planning algorithm, a function bounding the algorithm's reasoning process and a plan-ranking function that ranks plans according to a model of the hearer's plan-related preferences. Cooperative Plan Identification uses *refinement search* [19] as a model for the hearer's plan reasoning process. Refinement search is a general characterization of the planning process as search through a space of plans. A refinement planning algorithm represents the space of plans that it searches using a directed graph; each node in the graph is a (possibly partial) plan. An arc from one node to the next indicates that the second node is a refinement of the first (that is, the plan associated with the second node is constructed by repairing some flaw present in the plan associated with the first node). In typical refinement search algorithms, the root node of the plan space graph is the empty plan containing just the initial state description and the list of goals that together specify the planning problem. Nodes in the interior of the graph correspond to partial plans and leaf nodes in the graph are identified with completed plans (solutions to the planning problem) or plans that cannot be further refined due for instance, to inconsistencies within the plans that the

algorithm cannot resolve.

In the CPI model, when the speaker provides $\mathcal{P}_{\mathcal{D}}$ as a description, the constraints in this description create a planning problem for the hearer; the problem contains a specification of the planning problem corresponding to the task at hand (assumed to be commonly known by the speaker and the hearer) but, unlike the initial nodes in typical refinement search algorithms, this node does not contain an empty plan body. Instead, it contains the partial plan characterized by the content of the description. In the cooperative plan identification approach, a refinement search planning algorithm is modified to accept as input a non-null partial plan whose contents correspond to $\mathcal{P}_{\mathcal{C}}$.

To complete a candidate description, the hearer model's planning algorithm builds a plan graph rooted at the partial plan. During this process, the hearer model employs a preferences function f_h to direct search through the space of plans. The function characterizes the plan-related preferences of the hearer, mapping partial plans onto the natural numbers; the less preferred a plan p is, the greater $f_h(p)$ will be. The planning system uses f_h to rank plans that appear in the fringe of the plan graph as it is being constructed; search proceeds in a best-first manner, the planner expanding those fringe nodes that are ranked most promising. The system designer is free to rank plans using any criteria and to compute that ranking using any mechanism. See [36,11,1,12,40] for examples of the representation, acquisition and use of plan-related user preferences.

Best-first construction of the plan graph continues until the hearer model indicates that the resource limits of the of the hearer's interpretation process have been exceeded. The hearer's plan reasoning limitations are represented by a boolean *resource bound function* that accepts as input a plan graph representing the space already explored during a refinement search. The function returns the value *true* if the plan graph exceeds the hearer's search limit and returns *false* if it does not. In this work, a counter-based limit mechanism is used to limit the maximum size of the plan graph to a fixed number of nodes. The function returns $false$ just when its input graph contains a number of

nodes equal to or greater than the number of allowable refinements. Alternative definitions that take into account other aspects of the plan graph (such as the type of each refinement used to create a new node or the size of the plans being refined) may also be used.

When the hearer model's planning process halts, the resulting plan graph represents the space of all plans that the hearer will consider when interpreting the candidate description at its root. The complete plans along the fringe of the graph indicate all the solutions that the hearer model will consider; the subset of the graph's solutions that receive the lowest-valued ranking by f_h (i.e., that are most preferred by the hearer model) make up the graph's *preference set*. The preference set contains just those plans that the hearer model indicates may be adopted by the hearer upon hearing the current candidate description.

There are several ways that a change in the content of a plan description can alter the plan graph rooted at it and affect the candidate's suitability. For instance, removing detail from a candidate has the effect of creating a plan graph in which the root node is located farther away from the leaf nodes in the plan space. Moving the root node far from the leaves of the space may move the solutions to the problem beyond the resource bounds of the hearer model. Moving the root node closer to a solution by adding more detail to a candidate may so constrain the planning problem that alternative solutions within the plan space (but not included in the resulting plan graph) may be ruled out needlessly. Replacing some of the contents of a candidate with different plan structure also affects the content and organization of nodes that appear in the resulting plan graph. These modifications and the way they effect the appropriateness of a candidate description are demonstrated in the examples that appear the following sections.

4.1.2 Using the Model to Evaluate a Candidate Description

In the CPI model, a candidate description is evaluated by considering the plans that a hearer might adopt based on it. This evaluation is performed relative to requirements imposed by the speaker. In general, it is rarely the case that

a speaker requires that a hearer adopt a plan that is identical in every detail to the one that the speaker is describing. There is typically some amount of variance that the speaker can tolerate between the plan he is describing and the plan the hearer subsequently will form. I refer to plans that the speaker would be willing for the hearer to adopt as the result of his plan description as being *acceptable* to the speaker. In general, there may be a number of plans closely related to a source plan which the speaker considers acceptable. The degree to which these closely related plans vary from the actual source plan is dependent upon the speaker's tolerance for variation.

More specifically, acceptability is defined in terms of a *speaker* model, also provided as input to the CPI evaluator, in which the speaker's preferences are captured using a plan ranking function f_s. The domain and range of f_s are defined similarly to those of f_h but the function's values reflect the speaker's preferences over the structure of the plans that the hearer might adopt. To measure a candidate's acceptability in this work, I use the difference between the value of the speaker's plan ranking function applied to the plan potentially adopted by the hearer and f_s's value when applied to the source plan $\mathcal{P}_S$. To represent the amount of deviation from $\mathcal{P}_S$ that a speaker will tolerate in the plan that a hearer adopts, the speaker model contains a variable δ, $\delta \in N$, called the speaker's *tolerance level*.

For a given source plan $\mathcal{P}_S$ and candidate description $\mathcal{P}_C$, the set of *acceptable* plans (or simply the *acceptance set*) in a plan graph G_h rooted at $\mathcal{P}_C$ contains just those solution plans in G_h that are ranked by f_s within δ of the ranking assigned to $\mathcal{P}_S$.

Definition 1 (Acceptability) *Let G_h be the plan graph constructed by the hearer model when completing some candidate description $\mathcal{P}_C$.*

The acceptance set of G_h contains precisely those solution plans P' in G_h such that $|f_s(\mathcal{P}_S) - f_s(P')| \leq \delta$.

A plan in G_h is acceptable *just when it is in the acceptance set of G_h.*

In the limiting case, a speaker's tolerance level may be so constraining that acceptability corresponds to identity, although acceptability may be a much weaker notion in many contexts.

For a given partial description, the speaker must determine if the plan that the hearer will settle upon is acceptable to him. However, the preference function defined in the hearer model may not single out one solution plan as the preferred one; the preference set may contain many plans. In this case, the hearer model is not strong enough to predict which plan the hearer will adopt. In general, the system must determine if each of the preferred plans in a plan graph is acceptable. To do this, the system considers all of the complete plans in the plan graph created by the hearer model and determines the contents of the graph's preference and acceptance sets. When a candidate description contains sufficient detail to identify the source plan or one reasonably close to it, the resulting preference set will be a subset of the acceptance set. When this occurs, the description is called *successful*.

There are several possibilities regarding a description's content and its successfulness. To label these possibilities, I borrow terminology from analogous possibilities regarding the informativeness of referring expressions, as defined by Passonneau [30]. A candidate plan description $\mathcal{P}_\mathcal{C}$ is *successful* just when the two following conditions hold:

- at least one complete plan P exists in the plan graph G_h rooted at $\mathcal{P}_\mathcal{C}$. I refer to this property of a description as *resource adequacy*.
- all complete plans in G_h that are preferred by the hearer model are also acceptable to the speaker. This property of a description is called *preference adequacy*.

When a candidate contains no more detail than is necessary for its success, it is termed *efficient*. Candidates that are successful but not efficient are *overspecified*.

4.1.3 A Sample Problem

In this section I examine three candidate descriptions for the same example planning problem and discuss the way that the cooperative plan identification evaluator is used to evaluate each. The planning problem that I examine involves using the popular on-line service America Online (AOL) to get in touch with me. In the AOL domain, there are four basic ways to get in touch with me: by sending me email, by sending me an instant message (causing a dialog box with your message to appear on my computer screen), by entering a chat room where you know I'm connected and talking to me there, and by posting a message to me on a message board (the AOL equivalent of a newsgroup) that you know I monitor regularly. [5]

The plan space for this planning problem is shown in Figure 3. The graph in this figure is rooted at the null plan that describes only the initial state (not being "in touch") and the goal state (being "in touch"). The leaf nodes in this graph are the solutions to the planning problem and are labeled in the figure with text giving a rough indication of their structure. Each node is also labeled with an integer used for reference; these numbers correspond to the order that the hearer model will add nodes to the plan graph based on its model of the hearer's plan references.

The planning algorithm used by the hearer model in this example (and in the remainder of this paper) is the planning algorithm DPOCL [42]. DPOCL plans represent both hierarchical and causal structure using several types of components. DPOCL plans contain *steps* representing the plan's actions. Because these steps contain variables, DPOCL plans contain *binding constraints* that relate steps' variables to objects in the task domain. *Ordering constraints* define a partial temporal order over the steps in a DPOCL plan, indicating

[5] For expository purposes, the examples in this section have been structured to eliminate planning activity needed to establish preconditions – only hierarchical planning is performed. The techniques I present here, however, are applicable to planning problems using more expressive representations (i.e., causal as well as decompositional structure). All of the task domains used in the empirical evaluation of this work (see Section 6) exploit the complete DPOCL action representation.

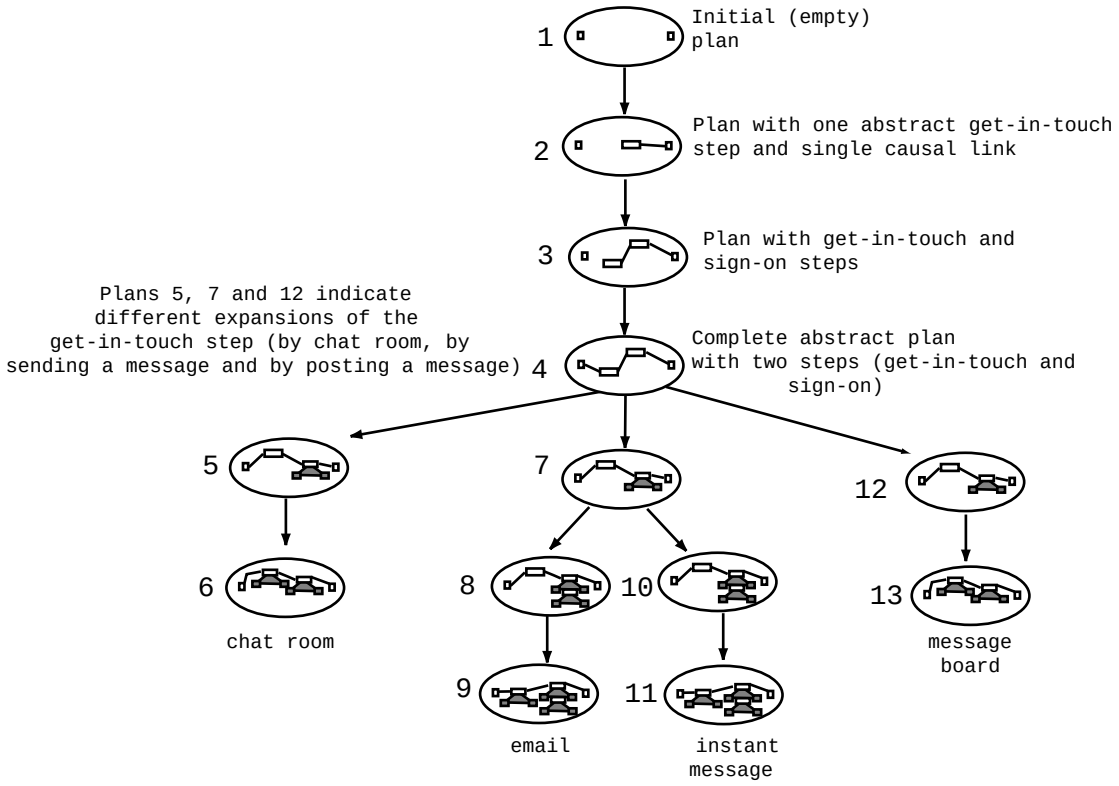

Fig. 3. Complete Plan Space for AOL Problem.

the order the steps must be executed in. Hierarchical structure in a DPOCL plan is represented by *decomposition links*: a decomposition link connects an abstract step to each of the steps in its immediate subplan. Finally, DPOCL plans contain *causal links* between pairs of steps. A causal link connects one step to another just when the first step has an effect that is used in the plan to establish a precondition of the second step.

The DPOCL algorithm performs refinement search as described in Section 4.1.1. Typically, refinement search planners iterate through a loop in which they first check to see if the plan space graph they've constructed contains a completed plan, then they perform plan refinement on the most promising nodes at the fringe of the current plan space. For DPOCL, plan refinement involves creating new nodes in the graph by repairing specific flaws in the plans associated with nodes on the graph's fringe. These flaws include steps with unestablished preconditions, steps with preconditions whose causal links are in conflict with

the effects of other steps in the plan, and abstract steps that have not yet had their subplans added to the plan's hierarchical structure.

To address the first type of plan flaw (a step with an open precondition), DPOCL creates a new plan by adding a causal link to the open precondition from a step whose effects can establish it. This link may be from a new step that DPOCL inserts into the plan or from a step that already exists within the old plan's structure. To address the second type of flaw (a causal link in conflict with another step in the plan), DPOCL creates a new plan by moving the conflicting step before or after the interval spanned by the causal link. To address the final type of flaw (an unexpanded abstract step), DPOCL creates a new plan by adding the hierarchical structure of the abstract step's subplan to the new plan as indicated by appropriate operators selected from DPOCL's operator library. The application of each method for addressing a particular flaw results in a new node being added to the plan space graph. In this manner, plans associated with nodes in the plan space graph become more *refined* the farther they are from the graph's root. If at any point DPOCL is unable to resolve a plan's flaws, that plan is marked as inconsistent and is pruned from the plan space. A complete description of the DPOCL planner and its plan representation can be found in [42].

I assume in the following example a limited resource bound on the hearer's plan reasoning, bounding the search she can perform to graphs with fewer than 5 nodes. This limit is artificially small to make for a more concise example – search limits used in the implementation described in Section 6 are considerably larger.

In this example, the speaker's source plan, $\mathcal{P}_S$, is the plan to get in touch by sending email (numbered #9 in Figure 3). I will assume that the speaker has two simple but strong factors that affect his plan preferences. For the purposes of my examples, the speaker will have a strong preference *against* any plan that involves either public communication or a violation of AOL etiquette. The speaker's own preferences and his amount of tolerance are set such that the only acceptable plans are those numbered #9 and #11 in Figure 3. The

other two complete plans in the figure, numbered #6 and #13, are both unacceptable given the speaker's preferences. Plan #6, getting in touch by talking in a chat room, violates the speaker's preferences for private communication. Similarly, plan #13, posting a message for me on a message board, violates this privacy preference. At the same time, the plan conflicts with the preference for obeying AOL etiquette, since personal messages are not appropriate as message board postings.

The following descriptions demonstrate the relationship between the amount of detail in a plan description and the plan space searched by a hearer model interpreting it.

Providing Too Much Detail Consider the following description:

Description 2: First, you need to sign on. To sign on, turn on your computer, monitor and modem. Find the triangular America Online icon among all your other software icons and double-click it to start the program. Choose the screen name you want to use (if you have more than one) by clicking the down arrow next to the Select Screen Name list box and clicking the name of your choice. Press Tab to move down to the Enter Password text box and type in your password. Click Sign On to open the connection to America Online.

To send a mail message, create a new mail message by clicking the Compose Mail button on the Toolbar. Type the recipient's screen name. Press Tab to move the blinking cursor into the CC: box. Enter the screen names of the people who should get a copy of the message but should not be listed as a main recipient. Press Tab again to put the cursor in the Subject box. Type a brief (32 characters or less) description of the message. Press Tab once more to get into the message area in the bottom of the screen. Type your message text in here. Click Send to mail the message. [6]

[6] The text in Description 2 comes from the instructions for logging into AOL and sending an email message in AOL found in [20]. The first six sentences (signing on to America Online) appear on pages 20-22; the next nine sentences (sending an email message to an America Online subscriber) appear on pages 110-112.

In this description, so much detail is provided that the description specifies precisely one completed plan: the source plan $\mathcal{P}_S$ that appears as a leaf node (#9) in the original plan space from Figure 3. The plan graph rooted at this node is shown in Figure 4.[7]

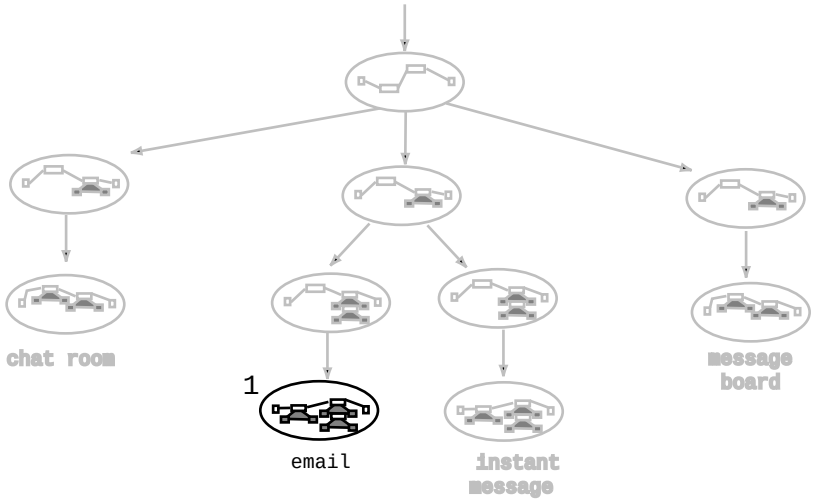

Fig. 4. Hearer Model's Plan Space for Description with Too Much Detail.

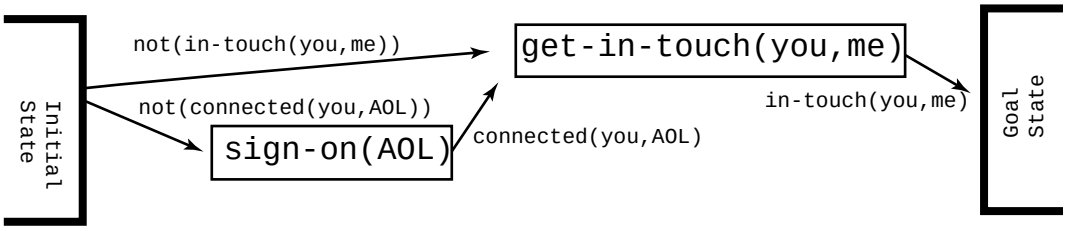

Fig. 5. Hearer Model's Complete Plan at Abstract Level (node 4 of Figure 3).

To evaluate this candidate description, the speaker uses the hearer model described in the previous section to complete this plan. Because the plan specified by the description is already complete, no search is required; this plan is the only plan in the set of preferred solutions specified by the hearer model. The plan is acceptable, since it is the source plan. The description demonstrates both resource and preference adequacy and is therefore a successful one.

[7] This figure and Figures 6 and 7 show plan graphs created by candidate descriptions. The graphs are shown in dark print embedded for reference within the original plan space, shown in a faint gray.

There is a problem with this description, however. By including so much detail in the description, the speaker eliminates another acceptable plan from consideration by the hearer – in this case the plan to get in touch by sending an instant message. This description is over-specified – moving the root node of the new plan space farther from the source plan would make for a more concise description while including other acceptable plans in the hearer model's preference set. However, as I show in the next section, this may also result in the inclusion of unacceptable plans in the preference set at the same time.

Providing Insufficient Detail Now consider the following description:

Description 3: Sign on to AOL and then get in touch with me.

This description describes a plan containing only the abstract SIGN-ON and GET-IN-TOUCH steps with no additional detail. The plan that corresponds to this description is shown in Figure 5; the plan graph rooted at this node is shown in Figure 6.

The order in which the hearer model will search this space is indicated by the integers labeling each node in the graph. With a search limit function constraining the hearer model's plan graph to contain no more than 5 nodes, the hearer model will only find one solution to the planning problem: node #3. Because the hearer model can find at least one completion, the candidate demonstrates resource adequacy. However, the completion that it finds is unacceptable given the speaker's preferences described earlier: node #3 involves *public* communication inside a chat room. There are, then, no plans in the preferred set of the hearer model that are also acceptable to the speaker. Consequently, Description 3 lacks preference adequacy and is therefore unsuccessful.

In this example, increasing the limit bounding the hearer models' plan reasoning would not correct the problem with Description 3. Even with a search bound that exceeded the number of nodes in the space in Figure 6, node #3

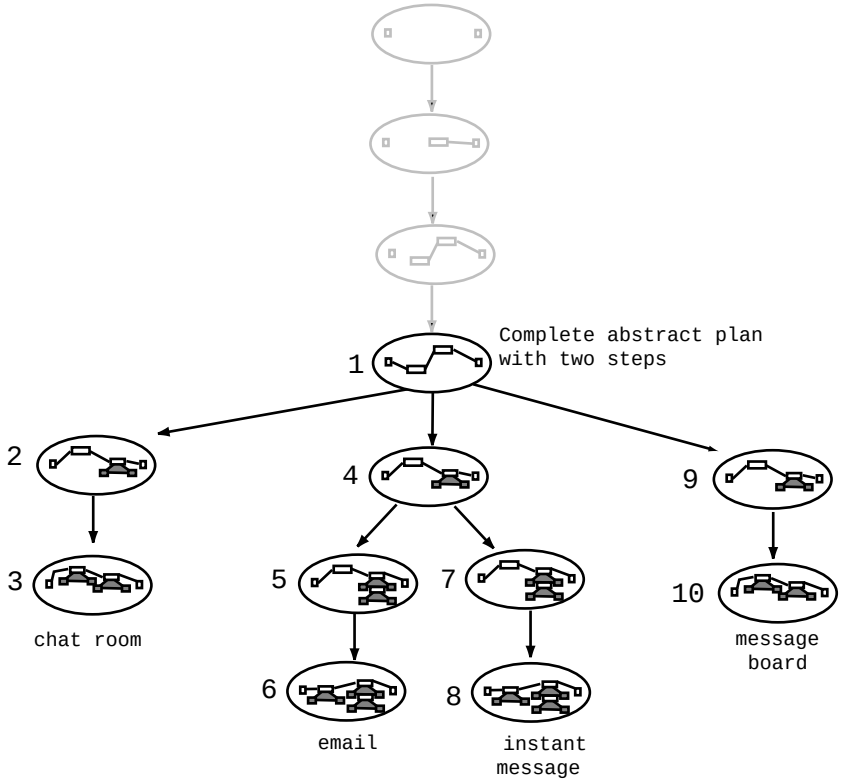

Fig. 6. Hearer Model's Plan Space for a Description with Too Little Detail.

remains in the preference set.

Providing Just Enough Detail The previous examples use descriptions that lie on either side of an effective, concise description for the source plan: Description 2 was over-specified while Description 3 was preference inadequate. One obvious technique for finding the minimal set of constraints that successfully describes the plan is to use a brute force search algorithm: considering every element in the power set of the constraints of the source plan (since effective descriptions may make reference to any or all of the components of a plan). This brute force technique begins its search by considering the initial, null plan and incrementally evaluates sets of constraints in a systematic manner, always considering the unexamined sets with smallest cardinality next. The algorithm halts when it either finds an acceptable plan or exhausts the power set of plan constraints.

Using this technique, it's possible to locate a set of plan constraints corresponding to the following description:

Description 4: First sign on to AOL and then get in touch with me by sending me a message.

This description describes a plan that contains the SIGN-ON step, the GET-IN-TOUCH step and a decomposition for that step involving a SEND-MESSAGE step. The plan is partial, since it does not specify how to sign on or which actions to use to send a message (by email or by instant message). The plan graph rooted at this plan is shown in Figure 7. The hearer model will search the plan space below this node and find two solutions to the planning problem: nodes #3 and #5. Both of these nodes are acceptable (in fact, they are the only two), making the description itself successful. Any other description of similar or lesser size would either be rooted at one of node #1's siblings or its parents (with plan graphs that would either contain unacceptable solutions in the preference sets or that would contain no solutions at all). Therefore, the description is efficient.

The technique of selecting minimal descriptions using exhaustive search corresponds to Dale and Reiter's full brevity interpretation of the maxim of Quantity used in the generation of referring expressions. The computational cost of this approach and several alternatives is discussed in the following section.

5 Generating Candidate Plan Descriptions

The examples in the previous section indicate the relationship between a plan description and the reasoning that the hearer will perform to interpret it. They also demonstrate the relationship between an optimum description and the plan graph rooted at it. The optimum plan description 1) is the root of a plan graph containing no unacceptable plans in the set of plans preferred by the hearer model and 2) contains the smallest number of plan components of all similarly qualified descriptions. Finding an optimal description is a difficult

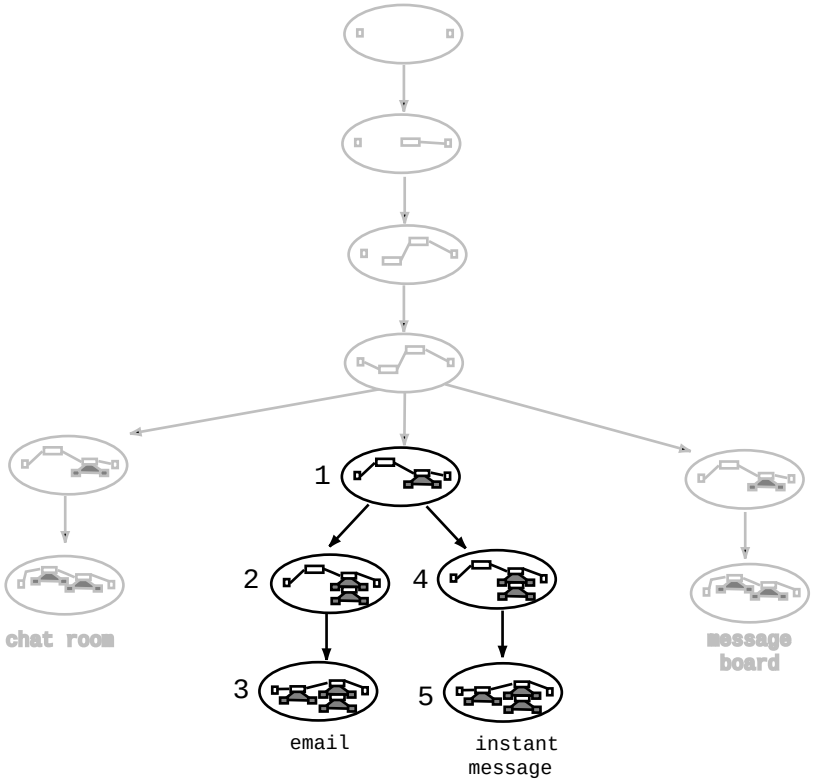

Fig. 7. Successful Plan Space.

problem since there is no obvious technique for constructing the description directly from the source plan. While exhaustive search through the space of all the possible candidates (the technique mentioned in the previous section) is guaranteed to find an optimum description, the computational cost of this search is prohibitive; since every coherent subset of the source plan could potentially serve as the plan's description, the space of candidate descriptions is essentially the power set of the set of components in the source plan.

Fortunately for the system implementer, there is evidence that optimal plan descriptions are not required for natural, concise and effective descriptions. As reported by Hull and Wright [41], people often generate non-optimal plan descriptions [8] and when they do so, their readers or hearers are still able to carry out the tasks at hand. A problem closely related to plan identification,

[8] This conclusion is also mirrored in Dale and Reiter's comment [5] on the prevalence of non-minimal referring expressions in naturally occurring discourse.

the task of generating referring expressions, is similarly constrained by efficiency limitations when algorithms search for optimal descriptions. Work in this area has adopted the strategy of restricting search to tractable subsets of the solution space in such a way that the systems generate concise (but not necessarily optimal) descriptions that are both natural and effective. This strategy has been adapted for use in plan identification and is discussed further below.

This section defines four implemented algorithms used to determine a set of plan constraints that will serve as a plan description. The first two are *cooperative techniques*, motivated by distinct computational interpretations of Grice's maxim of Quantity. These two algorithms serve as generator functions in implementations of the cooperative plan identification architecture. The second pair of algorithms represent approaches that do not take a model of the hearer into account, using instead two distinct techniques that directly translate the source plan into its description. These two *direct translation techniques* are used in the evaluation described in Section 6 in order to provide a basis for comparison against the two cooperative techniques.

5.1 Local Brevity: Exploiting a Plan's Structural Information

The *Local Brevity* algorithm searches for acceptable plan descriptions by moving from complete, detailed candidates toward partial, abstract ones. In this manner, the algorithm is similar to Dale and Reiter's Local Brevity algorithm; as in their approach, the CPI Local Brevity generator begins its search with a complete description (a complete plan) and creates new candidates by iteratively removing single components from the description based on local decisions dictated by a set of heuristics. The heuristics this algorithm uses are based on results from studies of the comprehension of instructional and narrative texts (referenced below) that indicate that the presence of some plan components in a plan's description are more important to the hearer's understanding of the plan than are others. To determine the order in which

components are deleted from the working description, the heuristics are used to assign a weight to each of the source plan's components. The algorithm iterates, first deleting the element in the plan that is weighted lowest, then passing the resulting working description to the evaluator function. Because the Local Brevity algorithm begins its search with a complete plan, the initial description is likely to be acceptable to the evaluator. The deletion process iterates until the evaluator indicates that the working plan has become too partial to be acceptable. At this point, the algorithm adds back in the last component that was deleted and uses the resulting description as the source plan's description.

The heuristics used to determine the order in which plan components are deleted are captured in two weighting functions, one used to rank plan steps and one used to rank causal links. These weighting functions each sum a number of terms representing the contribution of structural features of the plan to the importance of the component's appearance in the plan's description. Scaling factors are applied to each term, allowing relative weights to be adjusted easily.

A plan's steps are weighted based on three factors reflected in the three terms in Equation 1 below. The equation is motivated by the following heuristics suggested by the more qualitative results described in [33,34,37,14]:

- The greater the number of causal dependencies a step has on previous steps in the plan, the more important the appearance of the step is in the plan's description.
- The greater the number of subsequent steps that depend upon a step, the more important the appearance of the step is in the plan's description.
- The deeper a step appears in the plan hierarchy, the less important the appearance of the step is in the plan's description.

For a given step s in plan P, the weight w_s assigned to s is determined by the summation of three terms:

$$w_s = (\mathrm{In}(s,P) \times k_p) + (\mathrm{Out}(s,P) \times k_e) - (\mathrm{Depth}(s,P) \times k_d) \qquad (1)$$

In this equation, $\mathrm{In}(s, P)$ is a function returning the number of s's satisfied preconditions in P (i.e., the number of causal links leading in to s), k_p is a constant scaling factor for incoming causal links, $\mathrm{Out}(s)$ is a function returning the number of causal links leading out of s, k_e is a constant scaling factor for outgoing causal links, $\mathrm{Depth}(s)$ is a function returning the number of ancestors of s in P, and k_d is a constant scaling factor for step depth.

The values of the constant scaling factors are determined empirically and may vary between domains. All scaling factors in Equation 1 are constrained to be no less than 0.

A single factor is used to assign weights to the causal links in a plan description: links are weighted based on their relative temporal duration. Results from reading comprehension and text summarization studies [13,22,31] suggests that causal relationships between steps that are temporally close are often so readily reconstructed that references to the relationships are elided from plan descriptions. In causal link planners without an explicit representation of time (such as DPOCL), an estimate of the link's duration relative to other links in the plan can be made by counting the number of steps in the plan that might possibly occur between the link's source step and its destination step. The greater the number of intervening steps, the longer the duration of the causal link. In the CPI implementation, for a given causal link l from step s_i to step s_j in plan P, the weight assigned to l is expressed by the equation

$$w_l = (\mathrm{Inter(l, P)} \times k_l) \tag{2}$$

where $\mathrm{Inter}(l, P)$ is the number of all steps that could possibly intervene between s_i and s_j in P and k_l is a constant scaling factor for intervening steps. k_l is restricted to be no less than 0.

To determine the sequence of components to be eliminated from the source plan, both the components' weights and their position in the plan structure are considered. In general, components with lower weights are eliminated first. However, in order to preserve the decompositional structure of the partial

plan (in accordance with the constraint on referential coherence described in Section 3), steps are only eliminated from the leaves of the plan (that is, steps are only eliminated when they are either primitive steps or abstract steps whose children steps have already been eliminated). The elimination of causal links is not similarly constrained.

As steps and causal links are removed from the plan, all plan components that make reference to those steps and links are also eliminated. For instance, when a step is removed from a plan description, all causal links leading into or out of that step are removed, all ordering constraints for the step are taken out of the plan description and the step's binding constraints are also deleted.

5.2 The Plan Path Algorithm: Following the Source Planner

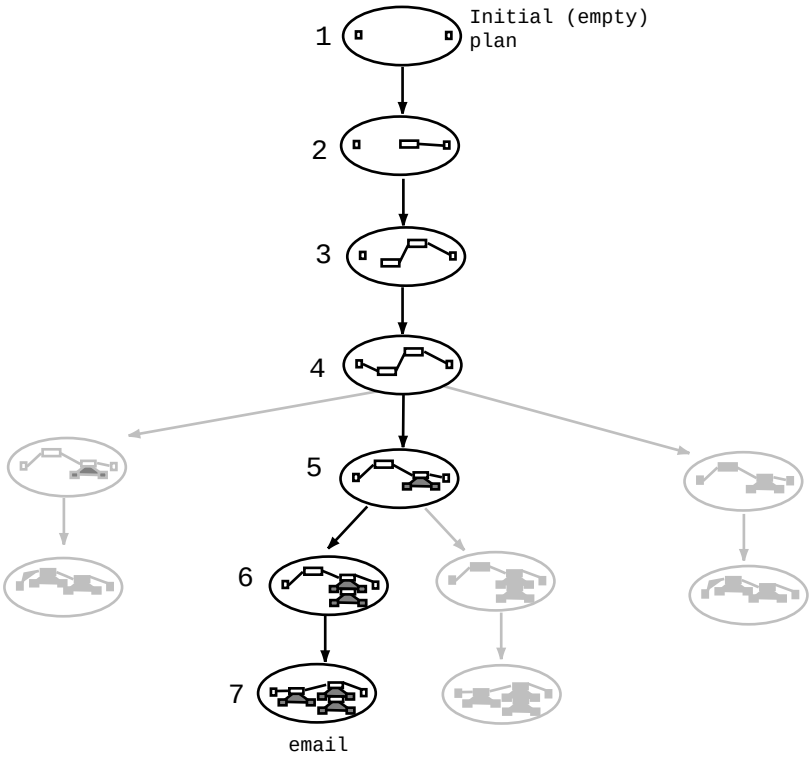

Fig. 8. The Sequence of Candidates Created By the Plan Path Algorithm in the AOL Domain.

The *Plan Path* algorithm generates candidate descriptions following a path through the space of plans created by the source planning system as it solved the original planning problem. The Plan Path algorithm begins its search by considering the null plan at the root of this graph and moves through the graph by selecting at each choice point the child node that lies along the shortest path from the root node to the source plan. When the algorithm visits a node in this space, it sends the partial plan associated with that node to the evaluator, testing to see if the plan can serve as a description. The Plan Path algorithm halts as soon as it finds a plan that is successful.

This algorithm requires access to the list of nodes that lie along the path from root to source plan in the source planner's plan graph. In the CPI implementation, the nodes are supplied as input by the source system along with the source plan. Providing these additional data structures is a minor requirement for a refinement planning system, since the nodes are created by the source planner during the search that produces the source plan to begin with.

An example of the series of candidate descriptions that the Plan Path algorithm creates is shown in Figure 8. This figure shows the plan graph for the AOL plan discussed in Section 4.1.3. The source plan (getting in touch with me via email) is labeled node #7 in the figure. The nodes in the graph that do not lie on a direct path from the root node (labeled node #1) to the source plan have been grayed out; the sequence of nodes that the Plan Path algorithm generates appears in the figure with plans in the sequence numbered in the order in which they are considered.

5.3 Two Direct Translation Algorithms

In order to provide comparisons to the cooperative plan identification algorithm, two direct translation approaches are also defined. The implementation of these approaches is described briefly below.

The Exhaustive Algorithm The component of Mellish and Evans's system that generated the content of a plan description was relatively straightforward: the system generated a description that referred to every component of the plan being described (with the exception of certain NONLIN bookkeeping structures). The same strategy for content selection is used here in the *Exhaustive* algorithm. To generate plan descriptions, the Exhaustive algorithm takes as input the source plan and, since every element of the plan is to be included in the description, the process returns the complete source plan as its output.

The Primitive Algorithm One possible approach to describing a plan is to describe just the lowest-level steps in the plan – those that will actually be executed. These steps correspond to the primitive steps in a DPOCL plan, the leaf node steps in the source plan data structure. The *Primitive* algorithm takes as input the source plan and searches through the plan's steps, selecting as the plan description the primitive steps in the plan, returning those steps in a total temporal order consistent with the source plan's temporal constraints.

There are several features of this algorithm that make it an intuitively plausible candidate for the generation of plan descriptions. Using this approach, the structure of a plan description directly maps to the actions that a hearer will execute. This direct relationship results in a description that makes the activities of the task clear to the hearer and it completely specifies the order in which they are to be performed. The descriptions themselves are brief, since they lack the hierarchical, causal and ordering structure present in the plans themselves. The technique does not require the hearer to form a complicated representation of the plan that the speaker is describing nor does the technique produce descriptions with gaps that the hearer must fill in before the description makes sense. To perform the task, it is sufficient that the hearer carry out the description's actions in sequence.

6 Empirical Evaluation

To evaluate the CPI model, I studied the empirical validity of the claim that providing conversational participants with a cooperative description of a plan increases the effectiveness of the communication. For this study, I followed Dixon [9] and defined the effectiveness of a description in terms of the answer to a specific performance-based question: Can the reader successfully carry out the task being described based on its description? In this experiment, human subjects were presented with a series of text descriptions whose content had been automatically generated by the four algorithms described in Section 5. The subjects were asked to carry out the plan descriptions in a simulated problem domain. Their actions were automatically recorded and this data was then analyzed along several dimensions to determine the quality of the subjects' performance. The hypothesis for the experiment stated that subjects that followed instructions produced by the cooperative techniques (i.e, the Local Brevity and Plan Path algorithms) would perform their tasks with fewer errors and achieve more of their top-level goals than subjects following instructions produced by the alternative approaches (i.e., the Exhaustive and Primitive algorithms).

The process used to produce the texts in this experiment is divided into three main components. The first module, consisting of the DPOCL planning algorithm, was used to construct solution plans for four planning problems in the task domain. The plans produced by DPOCL were then passed to the second component, a content determination module. For each input plan, this module applied each of the four approaches to content determination discussed in Section 5. The two generate-and-test approaches and the two alternative direct-translation algorithms each generated a corresponding plan description, resulting in a total of four descriptions for each input plan. Finally, the four plan descriptions were passed to a text realization module. The realization module determined the English text used to describe the plan components included in the descriptions as well as the order that the text appeared in the

text descriptions.

During the experiment, human subjects were asked to perform a series of four tasks — one for each of the four experimental source plans. For each task, I provided each subject with a list of the goals for each task (taken from the goal specification of the corresponding source plan) and a set of written instructions (one of the four text descriptions that had been produced for the source plan). They were asked to carry out the task as described by the text within a computer simulation. The simulation was constructed using the LambdaMOO text-based virtual reality system [4]. The task domain simulated a college campus and subjects' tasks involved running errands across campus (e.g., checking out books from the library, registering for classes at the Registrar's Office). Subjects interacted with the simulation via a command-line interface; the simulation was designed with a one-to-one correspondence between simulation commands and the primitive actions in the operator set used by the planner when creating the source plans for the experiment. The success of the subjects' activity in performing their tasks in the simulation was used as a measure of the effectiveness of the content determination technique that had produced the instructions that they followed.

6.1 The Textual Realization of Plan Descriptions

The problems of determining the organizational structure of a discourse and the syntactic constructs and lexical items used to communicate a plan's structure are considerable. Because this research deals only with the determination of the content to be included in a plan description, more complex issues in the syntactic and lexical realization of the description have been avoided.

To communicate a plan description to subjects in the experiment, the description's data structures must be translated into English in the form of instructional text. Consequently, descriptions of individual plan steps are realized by using imperatives. [9] While this work deals only with the generation

[9] See Paris and Scott [27] for a discussion of the role of other stylistic forms that

of texts containing imperatives, the techniques used to determine the content for a plan description are separate from text realization. The content determination techniques are applicable to the production of descriptions across genre.

To translate plan data structures into readable text descriptions of those plans, I used a straightforward procedure for textual realization (TR). In this experiment, each plan description was specified by a set of plan constraints composed from the five types of components in a DPOCL plan (as described in Section 4.1.3). These types are shown again in the following table, along with the kinds of phrases that might be used to describe them in an instructional context:

Component type	Example Plan Components	Text Description
Steps	CHECKOUT(?BOOK,?LOC,?ID),	Check out Canterbury Tales
Binding Constraints	(?BOOK = TALES) (?LOC = CIRC), (?ID = YOUR-ID)	from the circulation desk using your student ID.
Ordering Constraints	GO(REG-OFFICE) < SUBMIT(FORM1,REGISTRAR)	Go to the Registrar's Office, then submit your form to the Registrar.
Causal Links	PAY-FEES → REGISTER	Pay your fees so that you can register for classes.
Decomposition Links	WATCH(BRAVE) ↓ GET(BRAVE), LOAD(BRAVE,VCR) PLAY(VCR)	In order to watch Braveheart, get the Braveheart video, load the Braveheart video into your VCR and push the PLAY button on your VCR.

The process of textual realization for each plan constraint is fairly modular. Translation functions (written in Lisp) are used to translate a step's act-type

appear in instructions.

and its associated variable bindings into the corresponding imperative clause. Discourse markers and clausal structure that express temporal, causal and hierarchical relationships (e.g., "first," "second," "in order to") are used to convey information about the connective structure and ordering constraints in a plan.

To produce a text description, the TR process traverses the plan description data structure in a top-down order (respecting the temporal ordering of sibling steps). Text that describes an individual plan component is generated and added to the text description in the order in which the component is encountered during the traversal; as a result, the ordering of descriptions of individual plan components in the text follows the order that the corresponding data structures appear in the plan. This ordering strategy is similar to that used by Mellish and Evans and is consistent with Grosz and Sidner's [17] suggestion that the ordering of topics in a task-related discourse should follow the structure of the task itself. Further, this approach is consistent with studies of readers' comprehension of instructions which suggest that it is helpful to maintain congruency between the order in which procedural steps are mentioned and the order in which they should be executed [2,8]. [10]

To traverse the plan description, the TR procedure works from the top of the plan downward, describing the components of each subplan that it visits. A subplan's sibling steps are described together (including, in the initial case, the plan's top-level steps) in temporal order of execution. If sibling steps are only partially ordered, a consistent total ordering is arbitrarily chosen and the steps are presented in that order. Explicit temporal ordering constraints between sibling steps are conveyed via discourse markers along with the steps' description.

Once the steps in a subplan have been described, each step in the subplan is considered again, in the same order as before, in order to describe the

[10] Results for experiments in the production of face-to-face verbal instructions [41] indicate that this order is also followed, although not as strictly as is seen when written text is used.

step's incident causal links and its subplan (if it has one). First, all incident causal links are described. When more than one causal link is incident upon a step, the step's links are described in the order that the links' source steps appear in the plan description. Next, if the step is a composite step, the step's decomposition link and the subplan below it are described. To describe a parent step's subplan, first the decomposition relationship is explicitly marked via an *in order to* clause in which the parent step is referenced again, and then the traversal process is resumed, continuing with the steps in its subplan.

Example texts generated for plan descriptions used in this experiment are included in Appendix A.

6.2 Configuring the Experimental System

The system components described in the preceding section contain a number of user-specifiable parameters. The various settings for the parameters that were used in the experimental systems are described here. If no empirical evidence was available to suggest settings for the variables, the settings were selected to minimize their effect on the overall outcome of the experiment (in order to increase the dependency of the results on the CPI architecture and reduce the dependency of the results on the particular experimental hearer model). In all cases, the values for the settings were determined independent of and without regard to the experimental context. Where previous research results suggested particular values, or relative magnitudes for values, those results were adopted.

6.2.1 Local Brevity Weighting Functions

The Local Brevity algorithm uses weighting functions to assign weights to each step and causal link in a plan; the weighting functions appear in Equations 1 and 2 in Section 5.1. The values of the constant scaling factors that were used in the experiment are shown in Table 1. The research by Trabasso and Sperry and by Graesser *et al* discussed in Section 5.1 considers the factors expressed

by the terms in both of these equations relevant to the hearer's interpretation process. Determining actual values to assign to these variables based on the work of these psychologists is difficult. The empirical work does not provide quantitative measures of the roles that each factor plays in the results they report. Further, the original results are reported by different research groups using distinct methodologies and pursuing somewhat divergent goals.

The values for the scaling constants in Table 1 are assigned to reflect my estimation of the relative emphasis for each of the factors across the psychological results that I mention above. In that work, the role that any given action plays in establishing conditions needed by subsequent steps is the main factor that signals the importance of the step in a subject's cognitive model. This feature corresponds to the number of outgoing causal links in a DPOCL plan step, and so the scaling factor for outgoing causal links was given the highest weight in my experimental design. The amount of time between two steps was also central in the mental models of subjects, though not as prominent as the number of outgoing causal links. This feature, as described above, corresponds to the number of intervening steps in a DPOCL plan, and so the corresponding scaling factor was given the next greatest weight. Finally, two remaining two factors – the number of steps that establilsh conditions for a given step and the depth at which the step appears in a plan hierarchy – were given lowest weights. These constants are, of course, user-specifiable parameters and, short of performing extensive experiments that compare the performance of the system under various settings, no strong conclusions can be drawn about the *relative* merits of one set of values over any others.

6.2.2 The Hearer Model

The CPI hearer model contains three customizable parameters: the planning algorithm, the hearer's plan preferences and her plan reasoning resource limit. DPOCL, the planning algorithm described in Section 4.1.3, is used in the experiment as the model of the hearer's plan reasoning. The plan ranking function used in the experiment looks only at the size of the plan, preferring

Table 1
Experimental Values for Weighting Constants use in the Local Brevity Algorithm

Constant	Description	Value
k_p	incoming causal links	1
k_e	outgoing causal links	5
k_d	step depth	1

Step Weighting Constants and Their Values

Constant	Description	Value
k_l	intervening steps	2

Causal Link Weighting Constant and Its Value

short, hierarchically structured plans with few top-level steps. The definition of the search limit function used in the experiment is also straightforward. In the absence of empirical evidence to suggest specific values for hearers' plan reasoning resource bounds, an objective method was devised to automatically generate a setting for the limit used for each plan being described. Using this method, the mid-point is found between the greatest depth bound where a complete plan description is generated and the least depth bound where an empty plan description is generated. To compute the mid-point value, each algorithm's depth bound is initially set to 0. A plan description is generated using this depth bound setting, and the depth bound value is incremented until a plan description is generated that contains less than the complete structure of the source plan. This value is taken as the lower bound of the range for d. The process continues to iterate, incrementing the depth bound and producing a new description, until the description that is produced contains no detail at all. This value is taken as the upper bound of the range for d. The mid-point between the upper and lower bounds is then used as the depth bound when generating a description for that source plan. Although the use of this technique results in the assignment of depth bounds that vary between plans, the method provides a basis for comparison by defining the same relative point

Table 2
Experimental Design

Subject Group	PLAN A	PLAN B	PLAN C	PLAN D
S1	Ex	P	PP	LB
S2	LB	Ex	Pr	PP
S3	PP	LB	Ex	Pr
S4	Pr	PP	LB	Ex

LB = Text produced using the Local Brevity algorithm, PP = Text produced using the Plan Path algorithm, Ex = Text produced using the Exhaustive algorithm Pr = Text produced using the Primitive algorithm.

in the space of all candidates that each algorithm considers.

6.3 Design

Twenty-six subjects were used for the experiment, with the subjects drawn from a pool of paid participants (mostly University undergraduates). The experiment utilized a repeated measured design: subjects were randomly assigned to one of four subject groups and each subject in the experiment saw one version of each of the four items (i.e., the four pre-computed plan data structures). The four items and four subject groups were combined in a design shown in Figure 2.[11]

6.4 Procedure

Each subject participated individually in the experiment, with the experimenter present at all times during the experiment session. Each experimental

[11] In this figure, the subject groups (S1 through S4) are indicated down the leftmost column, the different source plans (PLAN A through PLAN D) are indicated across the topmost row. The algorithms used to produce textual descriptions are indicated by the elements of the internal cells of the table. The table indicates the texts that each subject group received as well as the order in which they received them.

session was divided into two parts. In the first part, lasting approximately an hour, the subject was trained in the use of the simulation. In the second part, each subject was given a series of four tasks to carry out within the simulation. The goals of each task corresponded to the goals of the corresponding source plan (as indicated in Table 2); the state of the simulation at the beginning of each task corresponded to the initial state of the corresponding source plan.

For each task, the subject was first given a handout that described the goals of the task and the current state of the world. Then the subject was presented with the text description of the corresponding source plan in the form of a set of instructions for achieving the task's goals. The text of the description, formatted as a single paragraph, was displayed on the simulation's screen. After the subject had read the text, the description was cleared from the screen and the subject began carrying out the instructions. The subject entered the command "DONE" when all the tasks described in the instructions had been completed or when he or she was unable, for any reason, to continue carrying out the instructions. When either 10 minutes had expired or the subject entered the DONE command, the current task was terminated.

6.5 Summary of Results

The data that was collected for each subject consisted of a series of four *executions*. Each execution represents all commands typed by the subject and all responses generated by the system during one of the four tasks the subject was presented with. Coding of the execution data involved a two-step process. First, subject commands containing command-line errors were flagged, removing them from further consideration in the analysis. [12] After these command errors were removed from the data, each subject command datum was marked to indicate a) the plan operator corresponding to the command that the datum contained, b) the number of preconditions that operator had and

[12] Flagged errors included commands that contained misspellings and typographical mistakes, commands that used improper syntax and lines that were not valid commands.

c) the number of preconditions for that operator that were not met in the simulation at the point that the command was issued. [13] Because of the one-to-one correspondence between elements of a command (i.e., command names, argument names) and the act-types, locations and objects in the simulation domain, this marking was straightforward.

To measure the success of the subject's execution, I used three dependent variables:

The Step Failure Ratio (SFAIL). The percentage of the total number of steps containing preconditions that failed during the execution.

The Precondition Failure Ratio (PFAIL). The mean percentage of the number of preconditions for a failed step that were unmet when the step was executed.

The Goal Failure Ratio (GFAIL). The percentage of the plan's top-level goals that were unachieved when the execution ended.

For each dependent variable, data was averaged over items (that is, over plans) and a two-way repeated-measures ANOVA was conducted. In order to determine if the experimental source plans themselves had an effect on subjects' performance, a separate analysis was performed for each item, using a one-way, between-subjects ANOVA [3]. Means and standard deviations for these variables, along with the results of the various analyses of variance, are shown in Tables B.1 through B.5 in Appendix B.

The patterns of means for each of the dependent variables clearly support the hypothesis that cooperative plan identification techniques (using the Local Brevity and Plan Path algorithms) produce more effective plan descriptions than the two alternative approaches (the Exhaustive and Primitive algorithms). In particular, the data indicated that the cooperative model has a significant effect on the number of execution errors performed by subjects

[13] When commands fail to execute in the simulation, it is because the state of the simulated world is such that one or more of the corresponding action's preconditions do not hold. Commands whose preconditions are all met always execute properly.

during tasks ($F(3,63) = 7.06$, $p < .05$), see Table B.1) as well as the number of goals left unachieved by subjects during tasks ($F(3,63) = 3.52$, $p < .05$, see Table B.3); the effect on PFAIL did not reach statistical significance ($F,3,63$) $= 2.60$, $p < .08$, see Table B.2).

Planned comparisons testing the specific prediction that the cooperative techniques produce more effective descriptions than the direct-translation techniques (Local Brevity and Plan Path *vs.* Exhaustive and Primitive) confirmed this hypothesis for both SFAIL and GFAIL. The comparisons also showed that the Local Brevity and Plan Path algorithms did not differ significantly from each other on any of the three dependent variables. See Table B.4 for all relevant F-values. This table indicates the pairwise relationships between the means for each of the conditions of the experiment.

To see whether one or more items (source plans) showed a different pattern from the overall results, I analyzed the data from each plan separately. Although there was no significant effect due to algorithm seen in any of the individual sub-analyses (F-values were less than or equal to 1.32 — non-significant), the patterns of the means were roughly similar to the overall patterns (see Table B.5).

In order to determine if the differences in the amount of detail contained in the the plan descriptions could have accounted for these results, I performed the analysis of variance for each of the dependent variables a second time, adjusting the number of errors for each subject by dividing the data by the number of components in the corresponding text description. Table B.6 shows the mean number of plan components per plan description for each of the four content determination algorithms. The table also shows the mean values for the three dependent variables for each algorithm (from Tables B.1 through B.3) divided by the corresponding algorithm's mean number of plan components per description.

The results of the second analyses of variance were the same as those reported for the initial analysis, with the following exceptions. The pairwise comparison

between Exhaustive and Plan Path algorithms indicates that the difference between the two on the measure PFAIL and SFAIL were no longer significant ($F(3,63) = 2.44$ and $F(3,63) = 2.57$, respectively — both non-significant).

6.6 Discussion

The data clearly show that when subjects follow instructions produced by the cooperative techniques, they make fewer execution errors and achieve more of their goals than subjects following instructions produced by the direct-translation techniques. Subjects' performance across all experimental variables shows an increase of roughly an order of magnitude, with statistically significant results obtained for both the step failure and goal failure ratios.

Since (almost all of) the results from the initial analysis are preserved when the data is adjusted to account for the length of the texts, the data suggest that the effectiveness of the cooperative techniques is not due simply to their more compact form. Not only do the cooperative approaches produce text of comparable or slightly longer length, on average, than the Primitive algorithm, but the Exhaustive algorithm means for all three dependent variables in the second analysis are lower than those for the Primitive algorithm. This suggests that the differences in the *content* of the descriptions produced by the techniques, specifically the additional structural information present in the Exhaustive algorithm texts, contributes to its performance.

The results do appear to be independent of the text realization technique. Among other requirements, the TR procedure was designed to produce texts of sufficient quality that the underlying activities they described were not obscured. Despite the limitations of the text, the data indicate that subjects were, in general, able to use the text to determine the underlying activities being described. The overall mean for execution errors was quite low (across all conditions, subjects committed execution errors at a rate of 6.8%), subjects performed most of their tasks without making a single error (65% of all executions were error-free) and they were able to achieve 100% of all task goals

in a high percentage of the executions (81% of all executions achieved all task goals).

Because of the apparent independence of the experimental results with respect to text quality, it does not seem likely that an improvement in the textual realization process alone would result in data contradicting the hypothesis that subjects' performance is higher when following instructions produced by the CPI architecture. While I suggested above that the results are independent of the particular plans and text production processes that were used, I do not claim that the results are independent from the various parameter settings used in the experiment. In fact, I suggest the opposite (although this notion was not evaluated by the present study). I believe that different settings for the system parameters (i.e., the hearer model and the weighting function) would certainly affect the results. In general, the more accurately the computational model reflects the reasoning performed by the hearer, the better the model's performance will be.

Rather than indicating a shortcoming of the approach, sensitivity to the parameter settings is one of the system's strengths. The parameterization allows system implementors to exploit any user and domain-specific knowledge – this experiment singles out one particular configuration of the system's parameters and compares its performance to alternative implementations rather than to alternative parameter configurations. Results of the current study indicate that the system achieves superior performance in the absence of strong domain and user-specific knowledge. More accurate settings for these parameters should only serve to strengthen the system's performance.

While the results of this experiment clearly indicate that the two cooperative approaches produce more effective instructions, the design of the experiment does not allow us to point directly at the reasons for their effectiveness. It is my sense that the reason for their superior performance is due to the correlation between the types of content that they produce and that content most effeciently used by people when understanding plan descriptions. The motivation for the design of these algorithms flows directly from cognitive models of plan

comprehension; the descriptions that they produce are intended to provide the core structure needed by subjects as the basis for re-creating plans.

Rather than simply providing the correct amount of detail in a plan description, the cooperative techniques also create descriptions containing the correct *type* of detail. A partial plan description poses a re-construction problem for the hearer, and the descriptions produced by the cooperative techniques most effectively assist the hearer in solving that problem. In cooperative descriptions, the important peices of the puzzle are explicitly provided to the user and additional pieces, detail that might obscure the description's meaning, are left out.

Both the plan path and local brevity algorithms work by selecting compact descriptions of plans whose contents are intended to aid in the hearer's plan reconstruction process. In the local brevity approach, this selection process is directly informed by the work of psychologists, picking out individual plan components that play central roles in a hearer's model of the plan. In the plan path algorithm, the selection process mirrors the contruction process that created the plan being described – a hierarchically oriented state-space algorithm that may also correspond to techniques used by people when forming plans. For both these approaches, candidate descriptions are checked against an explicit model of user comprehension to validate their effectiveness.

7 Conclusions

This paper describes the generation of textual descriptions of complex activities, specifically the generation of concise descriptions of the plans produced by computer systems. While related research has recognized the importance of using plan data structures as the source for textual descriptions of action, work in this area has either produced systems that generate overly detailed text or systems that avoid detail by using a greatly restricted action representation. In this paper, I describe a model for generating concise descriptions of plans that is motivated by Grice's maxim of Quantity. In this model, the

hearer is seen as a collaborator in the description process; an explicit model of her interpretation process is used to anticipate the results of potential communication. To produce a cooperative plan description, the speaker refers to this hearer model to determine if a candidate description contains appropriate information.

The work defines a computational model of a hearer's plan reasoning capabilities and uses this model to select between competing candidate plan descriptions. The hearer model is described in terms of the hearer's planning algorithm, her individual plan preferences and the resource limits placed on her planning capabilities. In this model, the task of interpreting a partial plan description involves the reconstruction of plan's missing detail. The hearer model fills in this detail using plan reasoning similar to the type of plan reasoning that was used to construct the plan in the first place.

By viewing the hearer's interpretation of a partial description as the process of completing a partial plan, I have provided a formal account for the requirements of this task. The abstract model of plan identification was combined in a generate-and-test architecture called cooperative plan identification. In the CPI architecture, candidate descriptions are selected by the generator component from a subset of all possible descriptions for a source plan. The architecture's evaluator function simulates the hearer's interpretation of the candidate description and classifies the description as successful or not. The process iterates until a successful plan description is isolated.

I defined two algorithms that were used as generator functions in implementations of the CPI architecture. One, the Local Brevity algorithm, selects candidate descriptions based on the importance that the elements of a description hold for the hearer's comprehension of the description as indicated by psychological studies. The other, called the Plan Path algorithm, selects candidates based on the processing that was used to produce the source plan. For both implementations, a common evaluator function was defined that used a domain-independent planning algorithm as the hearer model's planning system and a domain-independent approach to configure the hearer model's other

components.

To characterize the efficacy of the two cooperative techniques relative to one another and relative to two alternative direct-translation algorithms, I performed a task-efficacy evaluation [39]. In this experiment, subjects that followed instructions produced by the CPI algorithms committed fewer execution errors and achieved more of their tasks' top-level goals than subjects following instructions produced by other techniques. The experimental results provide clear support for the greater efficacy of the cooperative techniques.

Acknowledgements

Support for this work was provided by the Office of Naval Research, Cognitive and Neural Sciences Division (Grant Number N00 014-91-J-1694) and from the DoD FY92 Augmentation of Awards for Science and Engineering Research (ASSERT). Support for the preparation of this manuscript was provided by ONR grant N-00014-96-1-1222.

The author thanks Johanna Moore, Martha Pollack, Rich Thomason, Barbara Di Eugenio and David Allbritton for many discussions about the material presented here, as well as the helpful comments of the anonymous reviewers. Thanks also to Katia Sycara for support during the preparation of this manuscript.

A Example Plan and Text Descriptions

The figures in this section show plan descriptions for one experimental task (named Task D) in a graphical representation. In these figures, the descriptions' steps are indicated by using the names of each step's act-type and variable bindings bounded by a rectangle. A plan's causal links are shown using solid line arcs drawn from the source step to the destination step. When more than one causal link leads from the same source to the same destination,

only one arc is drawn. Dashed arcs indicate decomposition links leading from a parent step to its children.

Temporal ordering in the figures is indicated in a rough left-to-right layout in the figure, although the relative ordering between steps of different subtrees is not indicated by the subtrees' position. Any orderings between subtrees can be reconstructed from the ordering described in the corresponding text descriptions.

Figures A.1 through A.4 show the plan descriptions produced for Plan D by the Exhaustive, Primitive, Local Brevity and Plan Path algorithms, respectively. In these figures, the plan structure that has been elided from the source plan is retained and shown in a faint gray. The text descriptions for the four plan descriptions are also provided in this section.

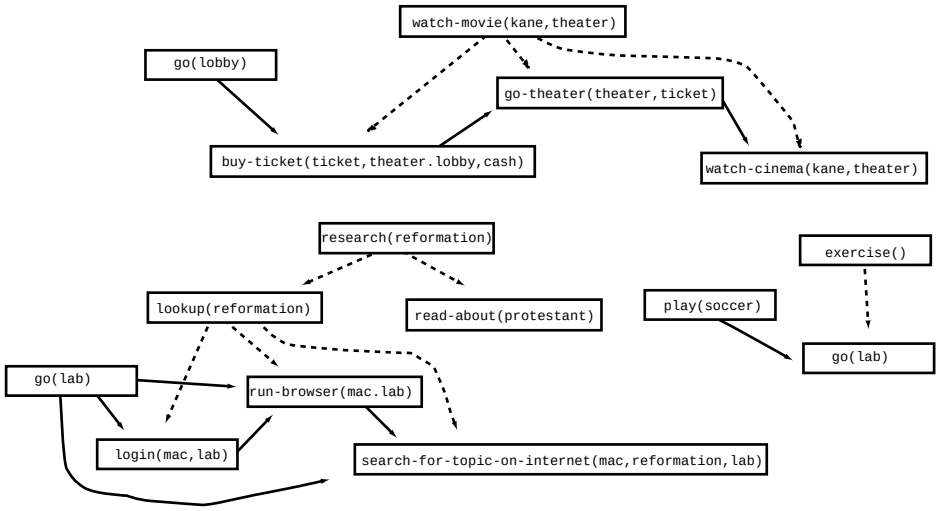

Fig. A.1. Plan D and its Exhaustive Description

Description 5 [Plan D, Exhaustive]: Go to the computer lab. Do some research on The Protestant Reformation. Go to the Fox Theater lobby. Watch Citizen Kane. Go to Hansen Field. Get some exercise. In order to research The Protestant Reformation, first look up The Protestant Reformation. Then, read the information on The Protestant Reformation. In order to look up the Reformation, first log in to the Macintosh computer

in the computer lab. Second, run a web browser on the Macintosh computer in the computer lab. Finally, search the internet for information on the Reformation using the Macintosh computer in the computer lab. Go to the computer lab so you can log in to the Macintosh computer. Go to the computer lab and log in to the Macintosh computer so you can run a web browser. Go to the computer lab and run a web browser so you can search the internet. Look up The Protestant Reformation so you can read the information on The Protestant Reformation. In order to watch Citizen Kane, first buy a movie ticket for the Fox theater at the Fox Theater lobby. Second, go to a movie at the Fox theater using a movie ticket. Finally, watch Citizen Kane at the Fox theater. Go to the Fox Theater lobby so you can buy a movie ticket. Buy a movie ticket so you can get in to a movie. Go to a movie so you can watch Citizen Kane. In order to get some exercise, play in a soccer game at Hansen Field. Go to Hansen Field so you can play soccer.

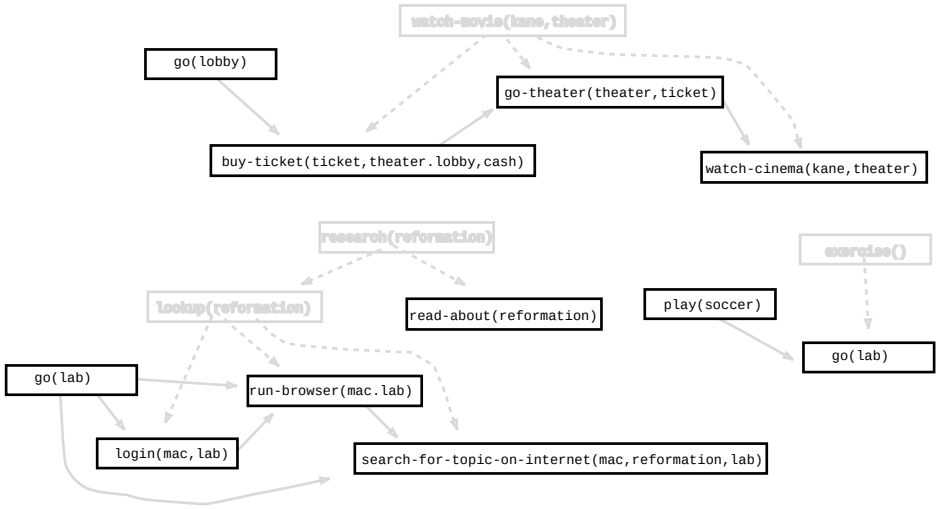

Fig. A.2. Plan D's Primitive Description

Description 6 [Plan D, Primitive]: Go to the computer lab. Log in to the Macintosh computer in the computer lab. Run a web browser on the Macintosh computer in the computer lab. Search the internet for information on The Protestant Reformation using the Macintosh computer in the

computer lab. Read the information on The Reformation. Go to the Fox Theater lobby. Buy a movie ticket for the Fox theater at the Fox Theater lobby. Go to a movie at the Fox theater using a movie ticket. Watch Citizen Kane at the Fox theater. Go to Hansen Field. Play in a soccer game at Hansen Field.

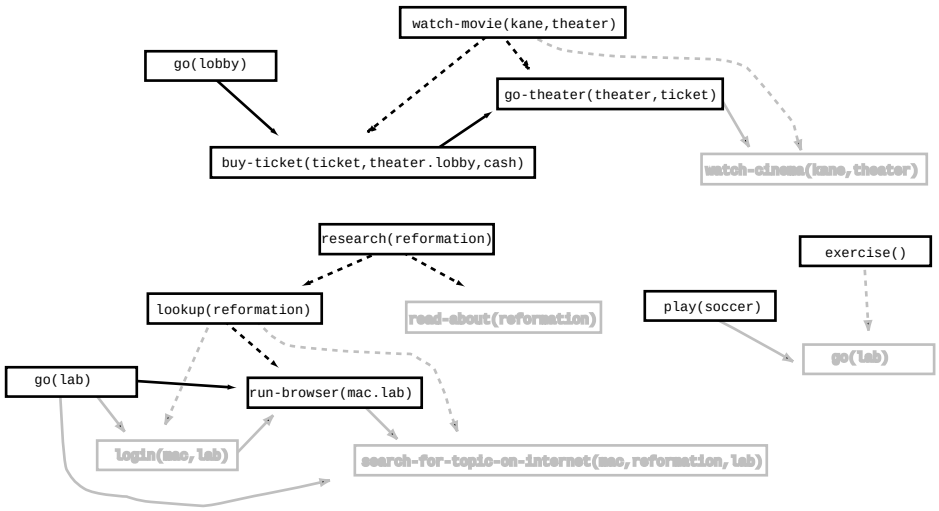

Fig. A.3. Plan D's Local Brevity Description

Description 7 [Plan D, Local Brevity]: Go to the computer lab. Do some research on The Protestant Reformation. Watch Citizen Kane. Go to Hansen Field. Get some exercise. In order to research The Protestant Reformation, look up The Protestant Reformation. In order to look up the Reformation, run a web browser on the Macintosh computer in the computer lab. In order to watch Citizen Kane, first buy a movie ticket for the Fox theater at the Fox Theater lobby. Then go to a movie at the Fox theater using a movie ticket. Buy a movie ticket so you can get in to the movie.

Description 8 [Plan D, Plan Path]: Go to the computer lab. Do some research on The Protestant Reformation. Watch Citizen Kane. Get some exercise. In order to research The Protestant Reformation, first look up The Protestant Reformation. Then, read the information on The Protestant Reformation. In order to look up The Reformation, first log in to the

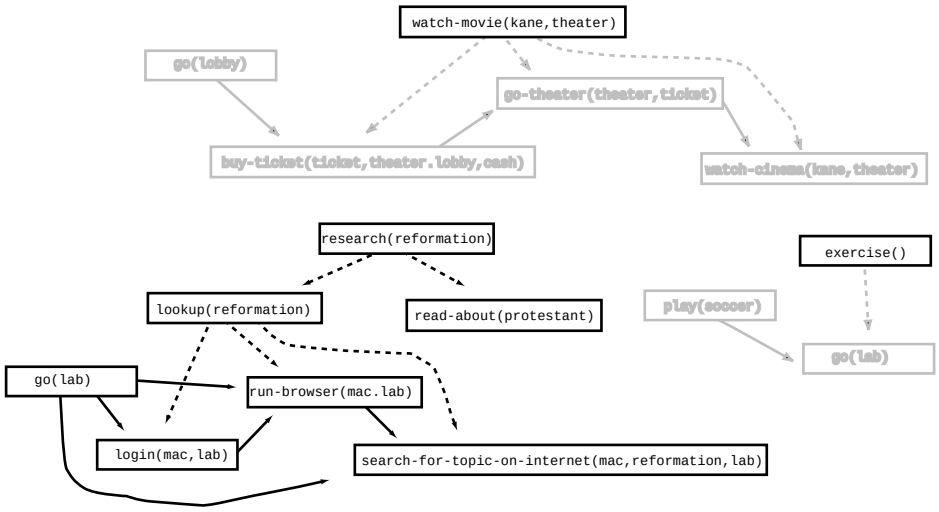

Fig. A.4. Plan D's Plan Path Description

Macintosh computer in the computer lab. Second, run a web browser on the Macintosh computer in the computer lab. Finally, search the internet for information on the Reformation using the Macintosh computer in the computer lab. Go to the computer lab so you can log in to the Macintosh computer. Go to the computer lab and log in to the Macintosh computer so you can run a web browser. Go to the computer lab and run a web browser so you can search the internet. Look up The Protestant Reformation so you can read the information on The Protestant Reformation.

These texts are not equivalent in quality to texts that would be produced by human writers describing the same content. However, in the context of the present study, this is not a limiting factor (as indicated by the data discussed in Section 6.6). The purpose of the text realization component was not to generate high-quality text but rather to provide a simple method for translating plan descriptions into sets of instructions that could be used in the evaluation process. Analysis of the experimental results, described earlier, confirms that the quality of the text did not interfere with the evaluation of the methods used to select each text's content.

Table B.1
Data for the Step Failure Ratio (SFAIL), the mean percentage of failed steps in an execution.

Exhaustive		Primitive		Local Brevity		Plan Path	
M	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*
12.88	12.90	7.89	11.13	2.32	6.57	2.47	8.59

Means and Standard Deviations for SFAIL

SOURCE	*df*	**SS**	**MS**	**F**
Algorithm	3	1846.86	615.62	7.06
Algorithm × Subject Group	9	999.18	111.02	1.27
Error(Algorithm)	63	5751.93	87.15	

ANOVA Summary Table for SFAIL

B Experimental Results

Tables B.1 through B.6 provide summaries of the experimental data discussed in Sections 6.5 and 6.6.

Table B.2
Data for Precondition Failure Ratio (PFAIL), the mean percentage of failed preconditions in each failed step of an execution.

Exhaustive		Primitive		Local Brevity		Plan Path	
M	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*
7.85	7.56	4.62	6.93	1.21	3.10	1.95	6.02

Means and Standard Deviations for PFAIL

SOURCE	*df*	**SS**	**MS**	**F**
Algorithm	3	258.84	86.28	2.60
Algorithm × Subject Group	12	344.11	28.68	0.89
Error(Algorithm)	63	2023.37	32.12	

ANOVA Summary Table for PFAIL

Table B.3
Data for Goal Failure Ratio (GFAIL), the mean percentage of unachieved goals in an execution.

Exhaustive		Primitive		Local Brevity		Plan Path	
M	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*
17.56	22.36	12.50	18.14	2.38	9.06	0	0

Means and Standard Deviations for GFAIL

SOURCE	*df*	**SS**	**MS**	**F**
Algorithm	3	1974.63	658.21	3.52
Algorithm × Subject Group	12	4243.74	353.65	1.89
Error(Algorithm)	63	11765.87	186.76	

ANOVA Summary Table for GFAIL

Table B.4
Contrast Analysis Showing Pairwise Comparison of Means

Technique	Mean	Technique	Mean	F Ratio
Exhaustive	12.88	Primitive	7.89	3.71
Exhaustive	12.88	Local Brevity	2.32	16.62*
Exhaustive	12.88	Plan Path	2.47	16.16*
Primitive	7.89	Local Brevity	2.32	4.63*
Primitive	7.89	Plan Path	2.47	4.88*
Local Brevity	2.32	Plan Path	2.47	0.00

Contrast Analysis for SFAIL

Technique	Mean	Technique	Mean	F Ratio
Exhaustive	7.85	Primitive	4.62	4.21*
Exhaustive	7.85	Local Brevity	1.21	17.85*
Exhaustive	7.85	Plan Path	1.95	14.08*
Primitive	4.62	Local Brevity	1.21	4.72*
Primitive	4.62	Plan Path	1.95	2.89
Local Brevity	1.21	Plan Path	1.95	0.22

Contrast Analysis for PFAIL

Technique	Mean	Technique	Mean	F Ratio
Exhaustive	17.56	Primitive	12.50	1.78
Exhaustive	17.56	Local Brevity	2.38	16.04*
Exhaustive	17.56	Plan Path	0.00	21.46*
Primitive	12.50	Local Brevity	2.38	7.13*
Primitive	12.50	Plan Path	0.00	10.88*
Local Brevity	2.38	Plan Path	0.00	0.30

Contrast Analysis for GFAIL

* indicates significant F-value.

Table B.5
Summary Means and Standard Deviations for Item Analyses

Plan	Exhaustive		Primitive		Local Brevity		Plan Path	
	M	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*
Plan A	10.19	12.34	8.69	9.99	0	0	0	0
Plan B	16.45	13.70	1.39	3.93	4.87	9.50	7.54	15.61
Plan C	8.78	11.20	9.38	18.75	4.42	7.57	0	0
Plan D	16.10	15.73	12.12	12.01	0	0	2.35	4.45

SFAIL Means and Standard Deviations by Plan

Plan	Exhaustive		Primitive		Local Brevity		Plan Path	
	M	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*
Plan A	6.59	7.40	5.47	7.12	0	0	0	0
Plan B	9.18	9.18	0.95	2.52	2.13	4.04	6.14	10.66
Plan C	6.98	8.04	4.29	8.57	2.71	4.16	0	0
Plan D	8.65	7.90	7.79	8.62	0	0	1.67	3.05

PFAIL Plan Means and Standard Deviations by Plan

Plan	Exhaustive		Primitive		Local Brevity		Plan Path	
	M	*SD*	*M*	*SD*	*M*	*SD*	*M*	*SD*
Plan A	10.71	19.67	15.48	20.65	0	0	0	0
Plan B	33.33	27.22	0	0	4.76	12.60	0	0
Plan C	21.43	26.73	8.33	16.67	4.76	12.60	0	0
Plan D	4.76	12.60	26.19	18.90	0	0	0	0

GFAIL Means and Standard Deviations by Plan

Table B.6
Dependent Variable Means Adjusted for Average Number of Plan Components

	Exhaustive	Primitive	Local Brevity	Plan Path
average number of components per description (ncomps)	52.5	26.5	32	26.25
SFAIL means divided by ncomps	.25	.30	.07	.09
GFAIL means divided by ncomps	.33	.47	.07	0
PFAIL means divided by ncomps	.15	.17	.04	.07

References

[1] M. Bauer. Quantitative modeling of user preferences for plan recognition. In *Proceedings of the Fourth International Conference on User Modeling*, pages 73–78, Hyannis, MA, 1994.

[2] H. Clark and E. Clark. Semantic distinctions and memory for complex sentences. *Quarterly Journal of Experimental Psychology*, 20:129–138, 1968.

[3] H. H. Clark. The language-as-fixed-effect fallacy: a critique of language statistics in psychological research. *Journal of Verbal Learning and Verbal Behavior*, 12:335–359, 1973.

[4] P. Curtis. Social phenomena in text-based virtual realities. In *Proceedings of the 1992 Conference on Directions and Implications of Advanced Computing*, Berkeley, CA, 1992.

[5] R. Dale and E. Reiter. Computational interpretations of the Gricean Maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263, 1995.

[6] J. Delin, A. Hartley, C. Paris, D. Scott, and K. Vander Linden. Expressing procedural relationships in multilingual instructions. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 61–70, Kennebunkport, ME, 1994.

[7] B. Di Eugenio. *Understanding Natural Language Instructions: A computational approach to purpose clauses.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, December 1993.

[8] P. Dixon. Plans and written directions for complex tasks. *Journal of Verbal Learning and Verbal Behavior*, 21:70–84, 1982.

[9] P. Dixon. Actions and procedural directions. In R. Tomlin, editor, *Coherence and Grounding and Discourse*. John Benjamins, Amsterdam, 1987.

[10] J. Donin, R. J. Bracewell, C. H. Frederiksen, and M. Dillinger. Students' strategies for writing instructions. *Written Communication*, 9(2):209–236, 1992.

[11] S. Elzer, J. Chu-Carroll, and S. Carberry. Recognizing and utilizing user preferences in collaborative consultation dialogues. In *Proceedings of the Fourth International Conference on User Modeling*, pages 19 – 24, Hyannis, MA, 1994.

[12] A. S. Gertner, B. L. Webber, and J. R. Clarke. Upholding the maxim of relevance during patient-centered activities. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pages 125–131, 1994.

[13] J.M. Golding, A.C. Graesser, and K.K. Millis. What makes a good answer to a question? testing a psychological model of question answering in the context of narrative text. *Discourse Processes*, 13:305–326, 1990.

[14] A.C. Graesser, S.P. Roberston, E. Lovelace, and D. Swineheart. Answers to why questions expose the organization of story plot and predict recall of actions. *Journal of Verbal Learning and Verbal Behavior*, 19:110–119, 1980.

[15] N. Green and S. Carberry. A hybrid reasoning model for indirect answers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 59–65, Los Cruces, NM, 1994.

[16] H. P. Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics III: Speech Acts*, pages 41–58. Academic Press, New York, NY, 1975.

[17] B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12:175–204, 1986.

[18] H. Horacek. Exploiting conversational implicature for generating concise explanations. In *Proceedings of the European Association for Computational Linguistics*, 1991.

[19] S. Kambhampati, C.A. Knoblock, and Y. Qiang. Planning as refinement search: a unified framework for evaluating design tradeoffs in partial-order planning. *Artificial Intelligence*, 76:167–238, 1995.

[20] J. Kaufeld. *America Online for Dummies*. IDG Books Worldwide, Foster City, CA, 1996.

[21] W. Kintsch. The role of knowledge in discourse comprehension: a construction-integration model. *Psychological Review*, 95:163–182, 1988.

[22] W. Kintsch and T. A. Van Dijk. Propositional and situational representations of text. *Psychological Review*, 85:363–394, 1978.

[23] C. Linde and J.A. Goguen. Structure of planning discourse. *Journal of Social and Biological Structure*, 1:219–251, 1978.

[24] D. L. Long and J. M. Golding. Superordinate goal inferences: Are they automatically generated during comprehension? *Discourse Processes*, 16:55–73, 1993.

[25] Mann and Thompson. Rhetorical structure theory: A theory of text organization. In Livia Polyani, editor, *The Structure of Discourse*. Abelbex Publishing Corporation, 1987.

[26] C. Mellish and R. Evans. Natural language generation from plans. *Computational Linguistics*, 15(4), 1989.

[27] C. Paris and D. Scott. Upper modelling: A level of semantics for natural language processing. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 54–61, Kennebunkport, ME, July 1994.

[28] C. Paris and K. Vander Linden. An interactive support tool for writing mulitlingual manuals. *IEEE Computer*, 29(7):49 – 56, 1996.

[29] C. Paris, K. Vander Linden, M. Fischer, A. Hartley, L. Pemberton, R. Power, and D. Scott. A support tool for writing mulitlingual instructions. In *Procedings of the International Joint Conference on Artificial Intelligence*, pages 1398–1404, August 1995.

[30] R. Passonneau. Integrating Gricean and attentional constraints. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1267–1273, Montreal, Canada, 1995.

[31] D. E. Rumelhart. Understanding and summarizing brief stories. In D. LaBerge and S. J. Samuels, editors, *Basic processes in reading: perception and comprehension*. Erlbaum, 1977.

[32] A. Tate. Generating project networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 888 – 893, Cambridge, MA, 1977.

[33] T. Trabasso and L. Sperry. Causal relatedness and importance of story events. *Journal of Memory and Language*, 24:595–611, 1985.

[34] T. Trabasso and P. van den Broeck. Causal thinking and the representation of narrative events. *Journal of Memory and Language*, 24:612–630, 1985.

[35] U.S. Robotics. *PalmPilot Professional Quick Start Guide*. U.S. Robotics, Mountain View, CA, 1997.

[36] P. van Beek. A model for generating better explanations. In *Proceedings of the Meeting of the Association for Computational Linguistics*, pages 215–220, Stanford, California, 1987. Association of Computational Linguistics.

[37] P. van den Broeck. The effects of causal relations and hierarchical position on the importance of story statements. *Journal of Memory and Language*, 27:1–22, 1988.

[38] K. Vander Linden and J. H. Martin. Expressing rhetorical relations in instructional text: A case study of the purpose relation. *Computational Linguistics*, 21:29–57, March 1995.

[39] M. Walker and J. Moore. Empirical studies in discourse. *Computational Linguistics*, 23(1):1–12, 1997.

[40] M. Williamson. *A Value-directed Approach to Planning.* PhD thesis, Department of Computer Science and Engineering, University of Washington, 1995.

[41] D. Wright and P. Hull. How people give verbal instructions. *Journal of Applied Cognitive Psychology*, 4:153–174, 1990.

[42] R. M. Young, M. E. Pollack, and J. D. Moore. Decomposition and causality in partial order planning. In *Proceedings of the Second International Conference on AI and Planning Systems*, pages 188–193, 1994.