A Computational Model of Collaboration on Reference in Direction-Giving Dialogues

by

Philip Glenny Edmonds

Department of Computer Science
University of Toronto
Toronto, Canada
October 1993

A thesis submitted in conformity with the requirements
for the degree of Master of Science at the
University of Toronto

Abstract

In a conversation, a speaker sometimes has to refer to an object that is not previously known to the hearer. This type of reference occurs frequently in dialogues where the speaker is giving directions to a particular place. To make a reference, the speaker attempts to build a description of the object that will allow the hearer to identify it when she later reaches it.

This thesis presents a computational model of how an agent collaborates on reference in direction-giving dialogues. Viewing language as goal-oriented behaviour, we encode route descriptions, referring expressions, and discourse actions in the planning paradigm. This allows an agent to construct plans that achieve communicative goals by means of surface speech actions, and to infer plans and goals from these actions. The basis is that a referring expression plan is acceptable to an agent if she is confident that the plan is adequate as an executable identification plan. By considering the salience of the features used in a referring expression plan, an agent can evaluate her confidence in its adequacy. Driven by the implicit intention of making plans mutually acceptable, the conversants collaborate until the hearer is confident in the adequacy of the current referring expression plan. In doing so, the conversants use suggestion and elaboration discourse actions that operate on the current plan. While collaborating, an agent is in a mental state that includes the intention to achieve the goal of having the direction recipient understand the directions, the plan the agents are currently considering, and a focus of attention into the plan. This collaborative state governs the discourse by sanctioning both the adoption of goals, and the mutual acceptance of plans. Reflecting the inherent symmetry in collaborative dialogue, the model can act as both speaker and hearer, and can play the roles of both the direction-giver and the recipient.

Acknowledgements

I thank my supervisor Graeme Hirst for his expert guidance and assistance. I also thank Jeffrey Siskind, my second reader, for providing many helpful comments.

I am grateful to Peter Heeman for the discussions that we had, for they helped in getting everything into perspective.

Finally, I thank my family and friends for bearing with me, and especially Lynne for her seemingly endless love and support.

Contents

List of Figures

Chapter 1

Introduction

Consider the following telephone conversation recorded by Psathas (1991, p. 196):

(1.1)	1	A.	Can you tell me where the Academy is?
	2	B.	Yeah, where ya coming from?
	3	A.	uh Newton.
	4	B.	Okay, why dontcha come up 128?
	5	A.	Yes.
	6	B.	And take 2A.
	7	A.	Yes,
	8	B.	um 2A will take ya right across Mass Avenoo an ya just stay on 2A, uh until ya get to Lowell Street.
	9	A.	Is it marked?
	10	B.	uh, Lowell Street?
	11	A.	Yeah.
	12	B.	Yeah I think there's a street sign there, its an intersection with lights.
	13	A.	Okay.
	14	B.	an ya turn right on Lowell Street. an its about quarter to half a mile um, take another right on Bartlett Avenoo.
	15	A.	Okay.
	16	B.	an that takes ya right to the Academy.
	17	A.	Okay.

Participant B is giving directions to participant A in response to A's query *Can you tell me where the Academy is?* This direction-giving dialogue contains many phenomena that we wish to discuss briefly.

First of all, A and B are involved in an interactive direction-giving dialogue. A is accepting each of B's direction instructions in turn with utterances such as *Yes* and *Okay*. When A does not understand an instruction, or does not find an instruction good enough, she puts the direction-giving sequence on hold and initiates a subdialogue to repair the offending instruction. For example, A is concerned that she will not be able to identify *Lowell Street* in utterance 8, so she initiates a subdialogue by asking *Is is marked?* in utterance 9. The subdialogue terminates when A utters *Okay* (utterance 13).

Second, B's direction instructions consist of an action (a verb such as *come up*, *take*, or *turn*), and, usually, a *reference* to a location or a road. The references to the places and paths are very important in making a good set of directions. But, in this dialogue, B is not using the common type of reference in which an agent attempts to identify the referent immediately. In fact, A is likely to have never even heard of the places and paths on which she will eventually travel. Appelt (1985c) calls this type of reference *nonshared concept activation with identification intention.* It is different from the type of reference that has been investigated in the literature, because the agents have no mutual knowledge of the intended referent, and the agent making the reference has the underlying intention for the other agent to be able to identify the referent eventually, given the information in the referring expression. This type of reference is pervasive in direction-giving dialogues and probably in all instructional dialogues.

Third, A and B are not just interacting to describe a set of directions, they are *collaborating.* Specifically, they collaborate to make referring actions successful. Returning to the subdialogue (utterances 9 to 13) above, we see that it serves to give more information about the location of the turn; but notice that both participants are making contributions to the referring action (or at least attempting to). A's utterance *Is it marked?* is more than a simple yes/no question. It is a suggestion to add the fact that *Lowell Street* is marked to the description, if possible. Additionally, this utterance also expresses that A is unsure about the reference to *Lowell Street* and requires more information. B responds to this suggestion (after resolving the ambiguous reference *it*) with utterance 12, by affirming that the street is marked with a sign, and by providing even more new information. B knew that he should provide extra information because he recognized that A was unsure of the reference from her utterance.

From the dialogue, we see that the participants are collaborating on more than just referring actions. They are collaborating to achieve the direction-giving successfully. After each utterance, A expresses her judgement, which allows B to know what to do next. The participants are mutually responsible for the outcome of the dialogue.

1.1 Goals

This thesis presents a computational model of how two agents involved in a direction-giving dialogue collaborate to make referring actions successful. Motivated by the type of referring action used in interactive direction-giving, and by the preliminary work of Appelt, our central goal is to model reference to objects for which the two agents have no mutual knowledge.

We develop a model that is highly influenced by Heeman and Hirst's (1992) model of referring, which is based on Clark and Wilkes-Gibbs's (1986). Since we view language as communicative action that people use to achieve their communicative goals, we use the planning paradigm to account for the construction and inference of referring expressions, route descriptions, and discourse moves.

For a referring action to be successful, we suggest that an agent must be confident that the referring expression plan is adequate as an identification plan for the referent. We describe our method of computing and evaluating confidence, which rests on the salience of the components of the expression. We discuss the importance of constructing salient referring expressions, and account for the behaviour with special actions in the plan schemas.

We model collaborative behaviour by using a set of meta-plans, or discourse actions, that operate on the referring expression plans. An agent can express judgement, refashion, or make suggestions by using discourse actions. Because an agent may find an expression inadequate, we define discourse actions for elaborating the expression that add more components to it. An agent may also want to suggest a good (salient) way to elaborate an expression, for which we also define a discourse action.

A second goal of this thesis is to model collaborative discourse where the domain plans are large (i.e., consisting of many subplans, subgoals, and effects). Since a domain plan may be large, we require a means of focusing on only part of it, so we extend Heeman and Hirst's notion of a collaborative state to include this focus of attention. The result is that the collaborative state sanctions the adoption of goals to express judgement about, refashion, or make suggestions for the subplan in focus, and sanctions the mutual acceptance of the subplan in focus, which allows its effects to occur (that update the common ground). The work of Clark and Schaefer (1989) on contributing to discourse was an influence for our model whose resulting discourse is similar to that seen in the real human dialogues studied by Clark and Schaefer.

A third goal is to provide an initial framework for describing routes by using a plan-based formalism that assumes that route description is a goal-oriented communicative task.

Reflecting the inherent symmetry in collaborative dialogue, we develop the model so that it can act as both the speaker and hearer. This means the model can play both the role of the agent who gives directions (and attempts to construct adequate referring expressions), and the role of the agent who listens to the directions (who attempts to understand the references).

Figure 1.1 is a diagram of the system and its knowledge sources. The system is divided into two main components. The first is for understanding a set of surface speech acts, and the second is for responding with a set of surface speech acts. The knowledge the system uses is also divided into two parts. Some knowledge is considered common knowledge, such

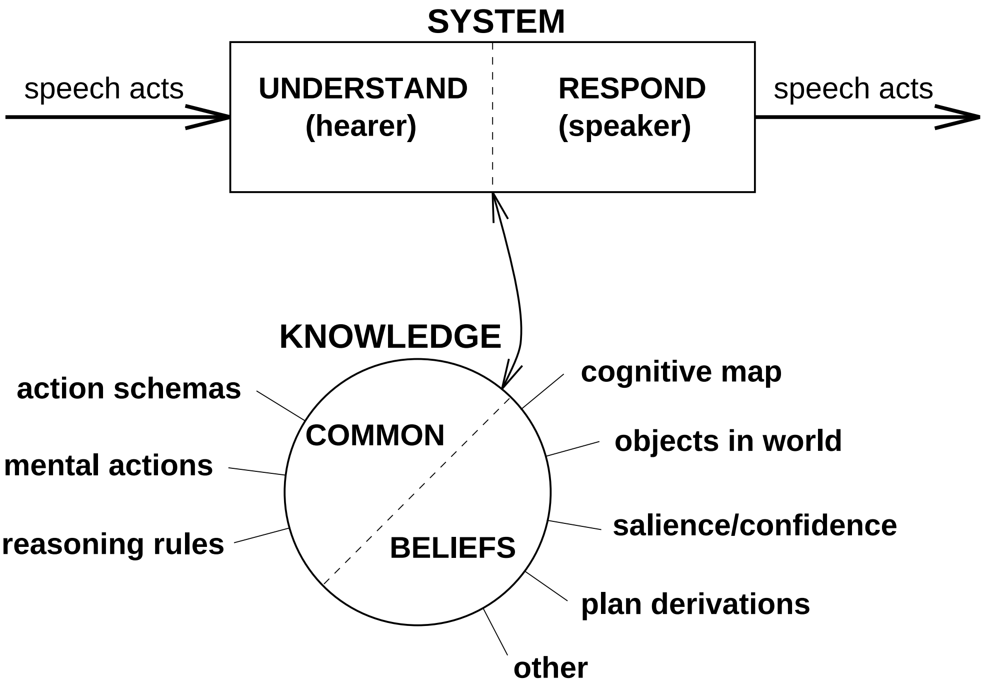

Figure 1.1: System diagram

as the plan schemas, mental actions, and reasoning rules. Other knowledge is modelled using beliefs, so that it can vary depending on the agent. Included in an agent's beliefs are his cognitive map, his beliefs about objects in the world (including their salience), beliefs about plan derivations and their validity, and other beliefs such as the belief that a suggestion was made.

This thesis has several contributions to make. First, we model a type of reference that has not yet been modelled. To do this we have to formalize intuitive notions about confidence and salience. Second, we amalgamate the work of Clark and Schaefer, Heeman and Hirst, and Grosz and Sidner (1986) to arrive at a plausible model of collaborative discourse. We model both the understanding and generation of language. And third, we give a preliminary account of how route description plans for interactive direction-giving can be processed.

1.2 Assumptions

We make a few broad assumptions to reduce the scope of our work.

First, we assume that the agents have mutual knowledge of the plan library, including route description plans, referring expression plans, discourse plans, and mental actions, and that they know the collaborative processes for reasoning about these plans. This is justifiable because referring expressions and discourse moves are communicative actions that everybody knows how to use (if they didn't, they couldn't communicate). As for route

```
1  A.  Go to the Lowell Street intersection.
       s-goto(Entity)
       s-refer(Entity)
       s-attrib(Entity,λX·category(X,intersection))
       s-attrib(Entity,λX·called(X,'Lowell Street'))
2  B.  Does it have a sign?
       s-accept(p1)
       s-accept(p1)
       s-postpone(p1)
       s-suggest(p1,[s-attrib-rel(Entity,Entity2,λX·λY·has(X,Y)),
                     s-refer(Entity2),
                     s-attrib(Entity2,λX·category(X,sign))])
3  A.  Yes, it does, and it also has traffic lights.
       s-affirm(p1,[s-attrib-rel(Entity,Entity2,λX·λY·has(X,Y)),
                    s-refer(Entity2),
                    s-attrib(Entity2,λX·category(X,sign))])
       s-actions(p1,[s-attrib-rel(Entity,Entity3,λX·λY·has(X,Y)),
                     s-refer(Entity3),
                     s-attrib(Entity3,λX·category(X,traffic-lights))])
4  B.  Okay.
       s-accept(p123) × 8
```

Figure 1.2: An example of the system's input and output

description plans, Psathas (1991) views direction-giving as a social task, so any agent who participates must be held accountable to a social contract, which means knowing how to generate and understand route descriptions.

Second, as shown in figure 1.1, the input and output of the system are both surface speech actions. We assume the existence of a generator and parser that translate between surface speech actions and natural language utterances. For example, in figure 1.2 we a show simplified version of part of dialogue (1.1), in which we have associated with each utterance its underlying surface speech actions. These actions are the actual input and output for our system.

Third, for the referring action, we make the strict assumption that the direction giver has complete (or sufficient) knowledge of any object that he refers to, and that the direction recipient has no specific knowledge of any referent at all (although she must have general knowledge about objects).

Finally, we only consider route description and not route planning. Furthermore, we take route description out of context and consider only the part of dialogue during which a route is described from origin to destination.

1.3 Some Terminology

It will be beneficial to have a few terms defined before we proceed. We will refer to the agents involved in a dialogue in a number of ways depending on their role in the current task. For direction-giving, the *giver* is the agent who plans a route description, and the *recipient* is the agent who attempts to understand it. While making references, the *initiator* is the agent who initiates the referring action (and is usually the giver in our domain), and the *responder* is the agent who will be identifying the referent, eventually (and, likewise, is usually the recipient). During a dialogue, the agent who constructs plans to achieve goals is the *speaker*, while the *hearer* attempts to understand utterances through plan inference.

When using pronouns to refer to the agents we adopt the convention that the giver, initiator, and speaker are masculine, and that the recipient, responder, and hearer are feminine. This distinction will help to clarify and disambiguate the text of the thesis.

Our planning terminology is fairly standard. We will use the terms *action schema*, *plan derivation*, *plan construction*, and *plan inference*. An *action schema* consists of a *header*, *where-clauses*, *constraints*, a *decomposition*, and an *effect*. An action schema encodes the constraints under which the header's effect can be achieved by the actions in the decomposition. We define three types of action in a decomposition: *primitive actions*, *mental actions*, and *action schema headers*. Primitive actions cannot be decomposed further. Mental actions are similar to primitive actions, but they have no observable effects. Instead, they affect the mental state of the planning agent. A plan derivation is an instance of an action schema that has all of the headers in its decomposition recursively expanded, and that has some, or all, of its variables instantiated. Plan construction is the process of finding a plan derivation that achieves a given effect through its set of primitive actions (its *yield*). Plan inference is the opposite process, whereby a plan derivation is found whose yield is a given set of primitive actions. In the text we will often refer to a plan derivation simply as a *plan*.

We will use many mental actions and propositions in the action schemas. A complete list, with definitions, is given in a glossary in appendix A.

1.4 Overview

Chapter 2 is an overview of previous research that is relevant to the thesis.

In Chapter 3 we describe our plan-based model of interactive route description. We describe how we represent the spatial knowledge of both topography and routes. We define action schemas for route description plans, and discuss how route description plans can be constructed, inferred, and evaluated. We give an example of how an agent constructs a route description plan. Route description plans provide a framework and motivation for our model of reference.

Chapters 4 and 5 are essential to tackling our goal of modelling collaboration on referring in route descriptions. In chapter 4 we examine the referring act in isolation. We first discuss how confidence in the adequacy of a referring expression can be assessed by considering salience. Then, we apply our formalism of confidence to the construction and inference of referring expression plans. We define referring action schemas, and go through an example of construction and inference.

In chapter 5, we show that the same planning processes can be used by the agents to collaboratively negotiate a referring expression plan. We discuss how an agent can express judgement about, make suggestions for, and elaborate a referring plan. We define the discourse action schemas that the agents use to perform these moves. We continue the example of the previous chapter to illustrate how the referring expression is made adequate.

In chapter 6, we complete our model of collaboration by tying together the route description and referring expression plans with an intentional structure that controls the flow of the discourse. We first describe the collaborative state that sanctions the application of reasoning rules for goal adoption and mutual acceptance. We describe how focus is maintained with a focus stack. We define goal adoption rules. We finally describe the acceptance process and the reasoning process, which complete our model of collaborative behaviour.

Chapter 7 presents a complete example of our system in action. We illustrate the construction, understanding, and negotiation of a route description plan with the system acting as both direction giver and recipient in turn.

Finally, in chapter 8, we summarize the contributions of our work, compare it to other related work, and speculate on possible future extensions.

Chapter 2

An Overview of Relevant Research

To achieve our goals we need to explore background material from a variety of disciplines including sociology, psychology, philosophy, linguistics, and, of course, computational linguistics. This chapter provides an overview of the relevant research. The first section discusses research about direction-giving and cognitive maps. The second section discusses speech act theory and referring expressions. In the third section, we discuss a model of collaborative discourse, and in the fourth, we discuss several theories of how planning can be applied to discourse modelling.

2.1 Direction-giving

Direction-giving happens in a variety of contexts. While we do not wish to model all types of direction-giving, it is important to investigate a few different direction-giving tasks, namely, understanding written directions, giving automated driving instructions, interactive direction-giving, and generating route descriptions. This section also includes a discussion of cognitive map theory.

2.1.1 Understanding Written Directions

Riesbeck (1980) addressed the natural language processing of texts of driving directions. He claims that, on a casual first reading, people make *clarity judgements* of the directions and do not use any complex spatial reasoning. This claim is intuitively plausible for several reasons. First, determining whether or not directions are clear and sensible (and possibly adequate) is of primary importance. Second, a clarity judgement is more simple, and therefore, less resource-consuming, than complex spatial reasoning such as map or route construction. Third, since the text will be available while making the trip, the spatial reasoning can be done then.

Riesbeck found three components in direction texts: *motions*, *descriptions*, and *com-*

ments. The most important of these is, of course, the motions which express commands or actions to be performed. They can take many surface forms as evidenced by the following examples:

> *Go south to Midway.*
> *Go two or three blocks.*
> *Turn right.*

Descriptions serve the purpose of telling what places look like, for example:

> *On the left is a gas station.*
> *A church is across the street.*

All other sentences are comments, such as

> *You can't miss it.*

which can imply motions or descriptions.

Riesbeck's system uses domain-specific heuristic rules based primarily on syntactic and lexical information, which he admits are crude but do the job. The system analyzes the text sequentially using rules that tie the language to conceptual forms (descriptions or motions). These concepts are then manipulated by *clarity* rules and *cruciality* rules until they are determined to be clear or not crucial. An example of a clarity rule is "left and right are clear", and of a cruciality rule is "motions are crucial."

One problem with this work is that the rules do not take semantic information into account. Directions are taken out of context, and analyzed separately (although inference over two sequential instructions is allowed). Riesbeck's argument that we do not need to construct mental maps to judge clarity may be too strong. Geographical knowledge is certainly important when giving clear directions and so could help with clarity judgements. While a mental map or a complete 'program' to execute the directions may not be necessary, a temporary local mental map (or route) would bring geographical knowledge into the judgement process.

There are times when spatial reasoning, including mental constructions of routes and maps, is necessary. One such occasion is when directions are given in a conversational context. In this task the judgement of clarity takes a seat beside actually remembering the details of the route.

2.1.2 Automated Driving Instructions

Another direction-giving task is that of giving real-time driving instructions in a non-interactive setting. Several systems have been developed that generate instructions to assist a driver to navigate in a city (Davis, 1989; Frisch et al., 1990; Cole et al., 1991). In these systems timing is critical, because instructions are given dynamically as the driver needs them at relevant points in a journey. This helps drivers, because they need not remember the instructions nor exert effort looking for a place to act. The dynamic nature of the system also allows for detailed instructions such as those necessary for navigating through large intersections.

The staff of the Laboratory for Computational Linguistics at the University of Illinois at Urbana-Champaign have developed a system that takes a previously computed route plan and dynamically generates natural language utterances for each 'move' (Frisch et al., 1990; Cole et al., 1991). Complex turns give drivers the most difficulty while navigating, so a move is defined to be from the end of one turn up to the next turn. The main components of the system include a driver action planner, a speech act planner, a sentence generator, and a map database.

The driver action planner does geometrical and functional analysis of the turns in the input route plan (a turn occurs between each pair of contiguous links in the route plan) and determines the action (or actions) necessary to navigate the turn successfully. Some sequences of turns may be viewed as single complex actions with sub-actions, and the planner identifies such sequences.

The speech act planner takes the possibly-complex driver action as input, and has the goal of getting the driver to perform this action. The planner uses a STRIPS-like approach with simple notions of intentionality—the driver's beliefs and intentions are modelled by STRIPS-style operators. For example, one premise is that the system can conclude that the driver knows that she needs to do something if she has been told and she is able to identify everything referred to by the system in its act of telling.

The system assumes copresence of the driver and the objects that need to be referred to in an instruction, since the driver is dynamically navigating through the environment. A further assumption is that enough information about the salient landmarks needed to describe locations is contained in a map database. So the speech act planner, by simply appealing to this database, can 'plan' the referring expressions that will uniquely identify the salient objects of a location. Unfortunately, the database contains ad hoc hard-coded information that cannot be tailored to the context, which may include variations in the knowledge and abilities of the driver, and the timing of the instructions.

In his Ph.D. thesis, Davis (1989) describes his Back Seat Driver system that tackles the same automated navigator task, but from a different perspective. He considers the current context to be important to give proper instructions. If the instruction is to be performed immediately, the system can refer to mutually observable information from the environment. But for instructions intended as cues, the places referred to are not yet observable by the driver. This means giving *clear* instructions that the driver has *confidence* in. A driver must be able to recognize the place to act, and must also believe that he will be able to recognize it. It is sufficient for the driver to have a plan that will uniquely identify the referent when it becomes observable even though the driver may not know of the place before getting there (Appelt, 1985a). But how can the system construct referring plans that a driver believes are adequate?

By using a simple hard-coded user-model, Davis's system can determine what the user believes is *salient*. Davis describes his salience hierarchy that ranges over landmarks: traffic lights, buildings, features of the road (underpasses, bridges, railroads, etc.), road 'endings', and street names. The hierarchy was obtained from empirical studies of how people understand and prefer given directions. When the more salient landmark types are used, instructions can be concise and a driver will be more confident in their adequacy. If the less salient landmarks are the only ones available then the driver usually requires longer and less concise instructions.

2.1.3 Interactive Direction-giving

Psathas, in several papers (Psathas and Kozloff, 1976; Psathas, 1986; Psathas, 1991), views the direction-giving task as an interactive social task that involves a *collaborative* effort between the two participants. Interactive direction-giving (i.e., during a conversation) is different from the other direction-giving tasks for several reasons. Here, interaction allows the participants to discuss the directions. It is no longer necessary to be as thorough when giving directions, either locally, as is the case with specific actions in automated driving directions, or globally by accounting for 'all contingencies' as good written directions should do. A further difference, as mentioned earlier, is that spatial reasoning is used when processing driving directions during conversation.

From his corpus, a set of telephone conversations involving direction-giving, Psathas identifies four aspects of the task: the opening, the sequence of operations (or directions that constitute a route plan), suspension and resumptions of the operation sequence, and the closing.

In the opening, an initial common ground is established, including the starting point, destination, mode of travel, time of travel, and mutual categorization[1] of the participants. Closings involve the direction giver marking the 'arrival' at the destination, and the recipient ending the direction-giving task by accepting this arrival point.

In conversation, as on paper, a set of directions is *structured sequentially.* Psathas observes that this structure is inherently embedded in a sequence of utterances spoken by

[1]The participants perform a type of user-modelling on each other to determine how much knowledge each other has. They determine membership in categories such as *local area person*, *stranger to the area*, or *visitor* (Psathas, 1991, p. 202).

the direction giver.[2] Participants take conversational turns in which the direction giver describes the next operation in the sequence and then waits for a display of acceptance or understanding by the recipient. The general pattern of the common operations in Psathas's corpus is that of

> **move to** a **reference point**,
> **at** which a **change** in direction is made.

So, a sequence of reference points is linked by operations performed in relation to them. These reference points are important to the recipient (because one doesn't want to miss a turn, or take the wrong turn), for whom they must be clear. When a reference point is not clear, the recipient must suspend the operation sequence. Alternatively, the giver may suspend the sequence to get confirmation that the recipient 'knows' a reference point. While the sequence is suspended, a side sequence[3] is initiated that serves to clarify the unclear reference point. Psathas does not dwell on the actual structure of the clarification side sequence, but he says in general it is a series of questions and answers initiated by either participant—a somewhat simplistic assessment. The side sequence is finished, and the direction-giving sequence resumed, when the recipient 'knows' the reference point.

The tendency in sociolinguistics is to not consider discourse pragmatics or computational models, and Psathas's work is not an exception. Nevertheless, he observes that the sequential nature of directions and direction-giving is reflected by the discourse structure and this is a good starting point for a computational theory. He provides evidence that the participants do collaborate, but he does not discuss a theory modelling that collaborative behaviour.

2.1.4 Generating Route Descriptions

Pattabhiraman and Cercone (1990) have implemented a prototype system that generates descriptions of bus route directions. They divide the task of *route communication* into the cognitive task of route finding and the linguistic task of route description, but make no claims about how these two tasks are interleaved. Even though route communication usually occurs in a dialogue, they are not concerned with the interactive aspects of the task: they model only generation and not understanding. Their text planner takes as input a *skeletal plan* and augments it by selecting from the knowledge base descriptions of landmarks, location, and orientation. The text planner has two components. The first

[2] Psathas observes this embedding from a sociological standpoint. Alternatively, the sequence of directions can be said to structure the sequence of utterances. The former view is useful when one is understanding a set of directions, while the latter is important for giving directions.

[3] A side sequence is a break in an activity that is not part of the main activity but is relevant. For example, a sequence of turns embedded between or within a contribution to a conversation is a side sequence (Jefferson, 1972).

augments the skeletal plan, and the second realizes the augmented plan as language by forming predicates and choosing utterance type. Thus, the text planner deals with both domain-level knowledge, and language-level knowledge.

They define the notions of salience and relevance, which are both used to control selection (and omission) in text planning. Salience is related to speaker-external objects and properties, and relevance to speaker-internal factors such as goals and intentions. Salience is the key factor in selecting landmarks to describe a location. Unfortunately, they present salience and relevance only in intuitive terms, but are working towards a concrete theory applicable in their domain. This work addresses the issues of route communication from the planning stages through to realization, and focuses on the generation of route descriptions. It does not treat route communication as a interactive goal-oriented task.

2.1.5 Cognitive Maps

Now that we have looked at the higher-level features of various direction-giving tasks, we must consider how route information can be represented for processing or for generating instructions. For a century, psychologists have studied cognitive maps. In particular, Lynch, a pioneer in this field, has done research on people's cognitive maps of cities; and Kuipers has developed a computational theory of cognitive maps of cities for use in spatial-reasoning tasks.

Kuipers (1978) explains that cognitive maps can be described metaphorically in three ways. First, the cognitive map is like many loosely related 'maps in the head'. Second, the cognitive map is like a network; specifically, it is a topological network of streets and intersections. Third, the cognitive map is like a catalog of routes, which are procedures for getting from one place to another.

Lynch (1960) discovered that people have five major elements in their cognitive maps: *paths*, *edges*, *districts*, *nodes*, and *landmarks*. The elements of the cognitive map are conceptual, and all exist together, interacting (overlapping, piercing, etc.) with each other. Although cognitive maps are topologically invariant with respect to the environment, they tend to be distorted in several ways: distances are compressed or extended, directions are twisted, etc.

The notion of salience comes up again, because elementhood can be marked on a scale of strength, where strength is related to salience and identifiability. Context determines how strong an element is, and even what type of element an object is. Contextual factors may include the point-of-view, familiarity, and current goals (i.e., to find a location, or to give directions) of a person. Salience can also be influenced by identifiability, visibility, prominence, and even functional importance (Devlin, 1976).

Sequences (of elements) are an important, possibly basic, sub-structure in a cognitive map that are especially useful for navigation because they facilitate recognition and memorization of a route. The majority of elements found in these sequences are paths and nodes, but landmarks are also prevalent. Strong elements are more likely to be used when describing spatial information linguistically. Therefore, strong paths, nodes, and landmarks are more likely to be used when giving directions.

Kuipers's TOUR model (Kuipers, 1978) is a model of commonsense knowledge of a large-scale urban environment. He is concerned with the everyday activities of learning (i.e., the assimilation of new observations into the cognitive map) and problem solving (i.e., the extraction of answers to particular questions of the cognitive map). One of his goals was to make the model as psychologically plausible as possible.

The elements and representations in the TOUR model are drawn loosely from those of Lynch, although Kuipers does not model the same primitives that Lynch describes. The model divides spatial knowledge into five primitive categories, the two most important being the *topological structure* and the *route*. The topological structure is composed of two sub-structures, *PATHs* and *PLACEs*, which are connected together by metric information (e.g., the angles of turns and travel distance) and topological information (e.g., the relative locations of places on paths, and the connectivity of paths). A route is stored as a sequence of *actions* (*TURN* and *GO-TO*) that, when executed, would take a traveller from one place to another. The fact that routes are modelled by actions makes it obvious that routes are the result of actual travel in the environment.

It should be evident that the theory is based on everyday action in the environment. Because information is assimilated by action, structures can be partial, with missing information (inference can be used to fill in the holes). In fact, the system maintains a current position pointer to help simulate traversals of the cognitive map as if the system were actually acting in the real environment.

The model is simple, and does not take input from linguistic sources, instead relying on an assumed conversion from visual input to internal representation. One way in which the model is not psychologically plausible is its reliance on metric information. Cognitive maps tend to be distorted and contain fuzzy information (Lynch, 1960; McDermott, 1980; McDermott and Davis, 1984). Although the processes of problem-solving (route-finding, and relative-position problems) that are manifest in his cognitive map are psychologically plausible, the map itself is not. A further problem is that landmarks are suspiciously absent from the theory. A route can only be described or followed on the basis of topological and metric information.

2.2 Language as Communicative Action

In the past few decades, philosophers have started to view language as intended and planned action. With this in mind, they went on to investigate the use of reference in spoken and written language. This section contains a brief explanation of speech act theory, and a discussion of the research on referring expressions.

2.2.1 Speech Acts

Speech act theory is generally considered to have been founded by Austin (1962) who analyzed utterances that he called performatives that did not express assertions about the world.[4] Searle (1969) went on to formalize the theory by defining *illocutionary speech acts* and *propositional speech acts*. At one level of abstraction, utterances are illocutionary acts such as requesting, informing, or promising. Central to the idea of illocutionary acts is that an agent makes an utterance with the intention that the hearer *recognize the agent's intention* in performing the act. Because the hearer can recognize the speaker's intention, she can infer the speaker's intended illocutionary act and change her mental state appropriately.[5]

At another level of abstraction an utterance can be viewed as a propositional act. Searle considers referring to be a propositional act because it is always part of an illocutionary act. We discuss this view below in section 2.2.2.

A further level of abstraction views utterances as *surface speech acts* (Appelt, 1985b). Surface speech acts are used to realize illocutionary acts and correspond directly to the syntactic structure of an utterance.

Since we are viewing language as communicative action, and since agents have intentions to achieve their communicative goals, we can use plan construction and inference techniques to account for these actions. Cohen and Perrault (1979) developed a system that would map an agent's goals to speech acts by constructing a plan, while, conversely, Allen and Perrault's (1980) system can infer a plan from its underlying surface speech acts. These plans can be incorporated into the larger domain of discourse planning, to be discussed after we discuss collaborative discourse models in section 2.3.

2.2.2 Referring Expressions

Referring as a Speech Act

Searle (1969) was the first to propose that referring can be treated as a speech act. Since he did not view referring as a solitary act (referring was only used within an illocutionary act), he had to consider it a propositional speech act. The propositional content consists of an identifying description that the speaker produces with the intention of uniquely identifying the referent.

Contrary to Searle, Cohen (1981; 1984) treats referring as an illocutionary act, explain-

[4]Before Austin, it was thought that all utterances expressed assertions, true or false, about the world.

[5]These changes are referred to as *perlocutionary effects*, and the speaker hopes that they are his intended effects.

ing that sometimes referring is done via a single separate utterance. A speaker uses a referring act to *request* that the hearer identify the perceptually accessible referent, and provides a description that the hearer decomposes into a plan to identify the referent.

Appelt and Kronfeld (Appelt, 1985b; Appelt and Kronfeld, 1987), like Searle, consider referring to be a propositional act used to realize an informing or requesting illocutionary act. The speaker's goal in uttering a referring expression is for the hearer to believe that it is mutually believed that the speaker and hearer have respective mental representations of the same object, the referent.

Heeman and Hirst (1992) also consider referring to be a speech act, but avoid the controversy (that it is an illocutionary or propositional act) by looking at the act of referring in isolation from other illocutionary speech acts.

Generating Referring Expressions

Appelt (1985a) considers generating referring expressions to be a primitive action in his sentence generation task. His algorithm first chooses a basic category descriptor for an object, then adds descriptors until the object is uniquely identified with respect to the mutual knowledge of the participants. Descriptors, based on facts about the object, are chosen if they are mutually believed by the participants, and if they are linguistically realizable. The aim is to generate an expression that the hearer can use to uniquely identify the object when it becomes observable.

Appelt's algorithm chooses descriptors indiscriminately. To make better expressions, Reiter and Dale (1992) improve on Appelt's process and on their own earlier work (Dale, 1987; Reiter, 1990). They examine psychological evidence and conclude that humans are more interested in making referring expressions that are easy to understand than they are in being efficient. The evidence shows that humans sometimes use unnecessary descriptors and are not as brief as possible. They propose several algorithms to account for this behaviour[6] but adopt an algorithm that chooses the most *preferred attribute* first (if it reduces the number of candidates). Preferred attributes are encoded in a list that usually includes visually salient attributes such as size, shape, and colour.

These algorithms are inadequate for use in conversation, because people don't always make an ideal or even adequate referring expression on the first attempt. Clark and Wilkes-Gibbs (1986) postulate three main reasons to account for this behaviour. First, time pressure may not allow an ideal referring expression to be designed. Second, the referring expression may be too complex to be presented in a single utterance. And third, ignorance of another's knowledge may cause a speaker to present an inadequate expression.

Because the above models of referring make strong assumptions about the beliefs of

[6] One algorithm would build referring expressions incrementally, with no backtracking, while another would use 'precompiled' reference scripts.

the participants they are not useful in conversation. Appelt assumes that the speaker has complete knowledge of the hearer's beliefs, and Reiter and Dale implicitly assume that the participants have identical beliefs (except that the speaker has knowledge of the attribute-value pairs that the hearer has no knowledge of).

Heeman and Hirst (1992) provide a more sophisticated model of belief. Each participant has their own beliefs about the properties of an object. The speaker must believe that the speaker and hearer mutually believe a property before using it in an expression. The hearer, in turn, might not have the same beliefs as the speaker, hence she can find the expression inadequate and initiate a repair process. Heeman, unlike Appelt and Searle, treats generation as part of the planning process—it is not a primitive action. In other respects, the algorithm is similar to Appelt's, but also draws from Reiter's earlier research on avoiding conversational implicature in generating referring expressions (Reiter, 1990).

Understanding Referring Expressions

Moving on to the opposite task, the understanding of referring expressions, the obvious approach is to treat the task as constraint satisfaction. The category descriptor gives an initial candidate set, and each component of the expression gives a constraint that serves to reduce the candidate set (Heeman and Hirst, 1992). Ideally this process would leave us with a unique candidate that is the referent. But, as stated above, the expression may be found invalid, a judgement that is made when either there are no candidates left (the expression is overconstrained), or several candidates left (the expression is underconstrained). It is these cases that cause the hearer to initiate a repair process to obtain clarification of the referring expression.

Copresence and Referring Expressions

Traditionally, researchers of referring expressions have assumed the following: that the agents have mutual knowledge[7] of the objects referred to (Appelt, 1985a; Appelt and Kronfeld, 1987; Heeman and Hirst, 1992; Searle, 1969), are copresent with these objects (Heeman and Hirst, 1992; Cohen, 1981), or have the objects in their focus of attention (Reiter and Dale, 1992; Grosz and Sidner, 1986). This type of reference corresponds to the speaker intending that the hearer either *know* the referent or *identify* it immediately, and to what Appelt (1985c) calls *shared concept activation with identification intention.*

[7]Clark and Marshall (1981, p. 17) recursively define two agents' mutual knowledge of a proposition p (that an object has a certain property) thus:

> A and B mutually know that p is true if
> (q) A and B know that p and that q.

Furthermore, a mutually known object is an object of which the speaker and hearer mutually know some properties (Appelt, 1985a, p. 1).

However, a speaker will commonly refer to objects that the hearer has no knowledge of. This type of reference occurs when a speaker wishes the hearer to be able to *identify* the referent at some later time. Appelt calls it *nonshared concept activation with identification intention*. For example, the instruction

Turn left at the third block past the stoplight

uttered to a hearer who has no knowledge of the location, carries the speaker's intention that the hearer identify (and be able to identify) the location at the appropriate time (Appelt, 1985c). Appelt says that

> the speaker's implicit intention that the hearer identify the referent may require the hearer to form and execute a complex plan to make the identification. Instead of planning a description with respect to the speaker and hearer's mutual knowledge, he tries to plan a description that is *useful* for the hearer to plan an identification action. (Appelt, 1985a, p. 2)

This type of reference occurs often in direction-giving dialogues, since Davis says "this description should be so clear that the driver cannot only recognize the place when it comes, but can also be *confident* in advance that she will be able to recognize the place" (Davis, 1989, p. 72). The problem with all of the aforementioned theories is that they do not address these two issues: (1) the speaker intending that the hearer eventually identify the referent, and so constructing a suitable referring expression (from which an identification plan can be built), and (2) the notion of confidence in the adequacy of this expression.

2.3 A Psychological Model of Collaborative Discourse

Since we are interested in how people collaborate when giving directions, we must first study models of discourse that account for the collaboration. Many researchers have investigated how people collaborate in discourse, and this section presents some of the psychological research. People communicate with one another so that they can interchange information (ideas, beliefs, knowledge, etc.). The result of a communication should be that the *common ground* of the participants is updated with the new information. A speaker in a discourse takes the common ground of the participants to be the set of his presuppositions, which are propositions whose truth he takes for granted. The common ground of the participants can also be described as their shared knowledge or *mutual knowledge* (Clark and Schaefer, 1989, p. 260). (See also footnote 7.)

First, we discuss Clark and Schaefer's model of how people contribute to discourse by using cooperative collective actions. And second, we discuss Clark and Wilkes-Gibbs's application of this model to the collaborative task of referring.

2.3.1 Contributing to Discourse

Clark and Schaefer (1989) discuss how people contribute to discourse. The basic units of conversation are *contributions* that result from the fundamental requirement that people must add to their common ground in an orderly way. That is, the content of every utterance is not automatically added to the common ground; the common ground only accumulates once the participants mutually establish that an utterance has been understood. A contribution is a *collective act* that establishes the mutual belief that the hearer understands what the speaker meant by his utterance (or action), so that they may then update their common ground.

Contributing divides conceptually into two phases:

> *Presentation Phase*: Speaker S presents utterance u for hearer H to consider. He does so on the assumption that, if H gives evidence e or stronger, he can believe that H understands what S means by u.
>
> *Acceptance Phase*: H accepts utterance u by giving evidence e' that she believes she understands what S means by u. She does so on the assumption that, once S registers evidence e', he will also believe that H understands.[8] (p. 265)

There are five main types of evidence that H can present to indicate her understanding of what S means by u. They are, arranged from weakest to strongest:

1. *continued attention*,
2. *initiation of the next relevant contribution*,
3. *acknowledgement*,
4. *demonstration*, and
5. *display*.

H must decide what type of evidence to present, and generally, the more complicated S's presentation, or the more demanding the current purpose, the stronger the evidence needed. Clark and Schaefer rely on several corpora, including the London-Lund corpus (Svartvik and Quirk, 1980), to determine empirically which types of evidence are expected for which presentations.

The acceptance process is inherently *recursive*, since giving evidence is itself a presentation that needs to be accepted (by presenting further evidence). But what keeps the recursion from going on ad infinitum? Clark and Schaefer propose their *strength of evidence principle*, which states that participants expect successively weaker evidence to be given. This principle implies that continued attention, or the initiation of the next turn, will end every acceptance phase, usually after one or two cycles of recursion.

[8] In giving evidence e', H intends that it is strong enough, that is, that S will find it equal or stronger than his expected evidence e.

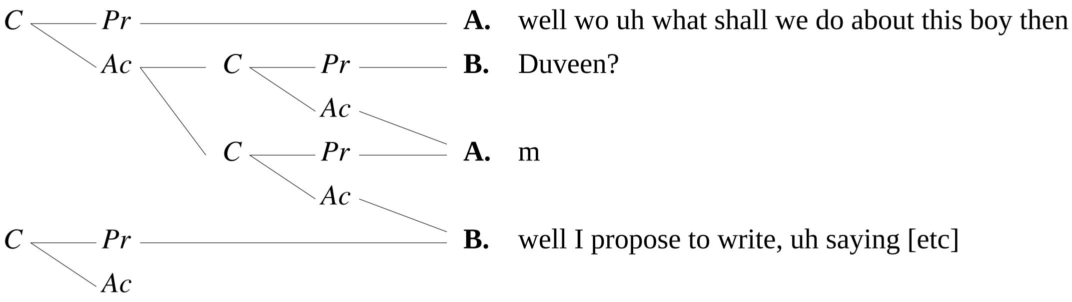

Figure 2.1: Hierarchical acceptance phase

Contributions can be embedded in many different structures, the simplest coinciding with the turn. Here, each turn is both an acceptance of the previous turn (completing the contribution), and a presentation that begins a new contribution.

More complex patterns occur when the presentation is not understood. In this case a hierarchical contribution structure within the acceptance phase arises. In dialogue (2.1) below, B is unclear about A's reference, and initiates a clarification sequence (indented) to clarify the reference to the boy, Duveen.

(2.1)	1	A.	well wo uh what shall we do about uh this boy then
	2	B.	Duveen?
	3	A.	m
	4	B.	well I propose to write, uh saying [etc] (p. 277)

Figure 2.1 shows the structure of this conversation. Note that a contribution C always has a presentation phase Pr and an acceptance phase Ac. The contribution begun by A, is finally accepted when B gives her answer, but the acceptance phase is a sequence that consists of two other simple contributions that coincide with the turns. This clarification subdialogue is a type of side sequence (Jefferson, 1972), and Clark and Schaefer found them to be pervasive in the corpora.

While the authors are not explicitly concerned with collaboration, their model can be the basis for a model of collaborative behaviour. Since people add to their common ground in an orderly way, they must cooperate, and this means using contributions. Contributions "are not formulated autonomously by the speaker according to some prior plan, but emerge as the contributor and partner act collectively. Success depends on the coordinated actions by the two of them" (p. 292).

Because people collaborate to achieve success in a conversation, they structure their contributions by using many different devices. For example, a side sequence during an acceptance phase allows the participants to focus on specific aspects of a presentation that they find troublesome. Or, by dividing his presentation into several installments, a speaker can give the hearer the opportunity to register information verbatim, or can assure himself that each step of his complex information is understood by the hearer before moving on. The size of contributions is also important in contributing effectively. Clark and Schaefer found that people preferred the surface form of their contributions to be one simple or complex sentence. They conjecture that the preferred contribution is one or more illocutionary speech acts.

What is missing from this structural view of collaboration is a formal theory of belief and intention to support it. The contribution model accounts for the overall structure of a discourse, but only informally explains the reasons an agent has for using a particular device at a particular time. In other words, they model the linguistic structure but not the intentional structure (Grosz and Sidner, 1986). The content of a contribution is left for another theory to explain. For example, in dialogue (2.1), B's utterance of *Duveen?* indicates that she has found A's reference to the boy inadequate, and initiates a clarification subdialogue. But this utterance is also a proposal to expand the reference by adding the boy's name. Furthermore, B is asking A if the expansion is correct (i.e., she is looking for confirmation that she has chosen the right boy). This example shows that there is more to collaborative discourse structure than just the embedded contributions—how does one make a clarification? Clark and Wilkes-Gibbs, in earlier work, delved into this matter.

2.3.2 Collaboration on Referring Expressions

Clark and Wilkes-Gibbs (1986) investigated how conversational participants collaborate in making a referring action successful. Consider a person A, who when speaking to a person B, refers to an object. For the referring action to be successful, A and B must establish the mutual belief that B has understood A's reference before moving on in the conversation. Clark and Wilkes-Gibbs argue that the participants make use of the acceptance process outlined above and that they use a number of inherently collaborative procedures to do so.

In a perfect world the acceptance process would be simple: A would be able to present the perfect referring expression which B could immediately accept. In their experiment, Clark and Wilkes-Gibbs discovered that the process is much more involved—subjects usually required several turns to complete the acceptance process. They propose that "participants aren't trying to ensure perfect understanding of each utterance, but only understanding to a criterion sufficient for current purposes" (p. 488). To initiate the process, A presents an initial version of a referring expression on which B passes judgement. B can either *accept* it, *reject* it, or *postpone* her decision until later. If B rejects or postpones then the expression must be refashioned by either A or B. Refashionings are accomplished in three main ways: *repairing* the expression by correcting speech errors, *expanding* the expression by adding more qualifications, and *replacing* the expression by a new expression.

The acceptance process continues, with each judgement/refashioning pair operating on and replacing the current referring expression (kept in the participants' common ground) until the expression is mutually accepted. Thus, the acceptance phase is either a single acceptance move, or a clarification subdialogue: a series of moves (judgements and refashionings) forming a side sequence.

Judgement and refashioning moves have their own acceptance process (separate from the referring expression's acceptance process); therefore there is a difference between finding a refashioning move unacceptable and finding its resulting expression unacceptable. Because the participants are collaborating, Clark and Wilkes-Gibbs propose that judgement and refashioning moves are always found acceptable (as long as they are understood), even though the referring expression that they operate on may be unacceptable. This view explains why the only dependence between succeeding judgement/refashioning pairs is the referring expression proposed by the former. Hence, the acceptance process for referring expressions is *iterative*, which has important ramifications on both the common ground of the participants, and the discourse structure (see Heeman and Hirst (1992, p. 27) for further explanation).

Collaboration can be defined in many ways. To explain their findings, Clark and Wilkes-Gibbs propose that participants assume *mutual responsibility* for the success of a referring action, and so cooperate with each other, each not impeding the other, and each presupposing certain beliefs. This principle of mutual responsibility embodies the notion that understanding need only be sufficient for current purposes. They also propose that, consistent with classical theories of least effort, the participants attempt to minimize their *collaborative effort*. There is a trade-off between initiating a referring expression and refashioning it, attributable to the three reasons discussed in section 2.2.2, which include time pressure, complexity of the referring expression, and ignorance of another's knowledge.

2.4 Discourse Planning

In this section we discuss a few methods that researchers have considered for applying planning to discourse modelling. First, we discuss how clarification subdialogues can be planned, and we describe Heeman and Hirst's computational model of Clark and Wilkes-Gibbs's theory. And second, we discuss a variety of models, each addressing specific issues, for discourse planning in task-oriented, interactive domains.

2.4.1 Clarification Subdialogues and Referring

Litman and Allen's (1987) approach to a plan-based discourse model involves having domain plans that encode topical knowledge of the conversation and discourse plans that encode the relations between utterances and the topic of conversation. The system uses plan recognition to infer a discourse plan from an observed utterance, from which it attempts to identify the underlying plan. The importance of this work is that meta-actions (meta-plans) are used to model discourse phenomena such as clarifications. They present three clarification plans that account for a variety of clarification subdialogues.

Heeman and Hirst (1992) render Clark and Wilkes-Gibbs's model computationally, by casting it into the planning paradigm (by extending Litman and Allen's model of clarification). To account for the collaborative behaviour of two agents, they propose that each agent is in a mental state which includes both the intention to achieve the main goal of the task (a successful referring action), and the current plan the agents are collectively considering that may achieve this goal. The collaborative state coordinates the agents' activities by sanctioning the adoption of goals to clarify (expressing judgement, and refashioning) the current plan, and by sanctioning the acceptance of plans that do so. Heeman and Hirst view plans as *mental objects* as opposed to data structures (Pollack, 1990), which allows an agent to have beliefs and intentions about them. Thus, an agent has a belief about the current plan's validity (whether or not it is mutually believed that the plan refers to a unique referent), and has the intention to make this belief mutually believed.

Both Litman and Allen's, and Heeman and Hirst's model are governed by discourse expectations, but the former uses a plan stack to remember the current context, whereas the latter incorporates the expectations into the rules (that implicitly model intentions). Because Heeman and Hirst use a collaborative state that contains a single current plan they are able to model satisfactorily how a plan is updated.

2.4.2 Collaborative Task-Oriented Dialogues

To apply his model to collaborative task-oriented dialogues, Heeman (1993) adapts it by providing a minimal set of operations, and their associated speech acts, that can be performed on the current plan. These operations include suggesting additions to the plan, requesting judgements, making judgements, and making replacements. The notion of a current plan is modified somewhat; the current plan now includes the proposed actions for solving the problem (and how they contribute to the main goal), and beliefs about the validity of the plan. Because the plan may contain many subplans, and because an agent may want to restrict his attention to just one of these (usually the one relevant to the latest addition), he introduces the notion of the *subplan in focus*, which the above operations take as a parameter. He does not provide a good account of how focus shifts, leaving this for future work.

Another model similar to Heeman and Hirst's is that of Lambert and Carberry (1991). The main difference is that they do not conflate problem-solving activities with communicative activities as the former model does, giving them a tripartite model. Their claim is that a model without this distinction cannot capture the relationship among several utterances that are part of the same higher-level discourse plan unless the first utterance determines what the higher-level plan is. It is unclear that this distinction is important. Heeman and Hirst claim that their model is more simple because both activities are embodied at one level.

Chu (1993) uses Lambert and Carberry's model as a basis for a model of collaborative discourse in which the system and user might disagree on a proposed action for an existing plan. Her approach accounts for, from the system's point of view, how a disagreement can be detected and negotiated, and how the existing plan can be modified. She separates the dialogue model into two parts: the existing model containing the current shared plan (that is mutually acceptable), and the proposed model containing the actions inferred from the last utterance that have not yet been mutually accepted. The system and user negotiate using collaborative meta-plans to correct the errors in the proposed actions. Once correct, the proposed actions update the existing shared plan. This model is similar to Heeman and Hirst's because it models collaboration with a set of meta-plans that are used in a subdialogue to correct a plan. The two parts of the dialogue model, when taken together, are like Heeman's current plan (that is kept in the common ground), but Chu assumes that the existing model can never be rendered invalid by the addition of a proposal, and so, it never requires re-evaluation. In Heeman and Hirst's model system there is no distinction between existing and proposed actions, so the current plan is constantly evolving.

Carletta (1990a; 1990b) addresses repair and replanning in an interactive task. In the task, two agents are given a 'map' containing some objects. The first agent is also given a route that she must describe to the second agent. The system plays the role of the first agent and so constructs the initial plan on which all repairs are done. Carletta uses a STRIPS-like approach with post-requisites to check for plan failure. Like the aforementioned models, the system uses meta-actions to encode the repair of referring expressions within the plan (all knowledge is mutually known, so it is only the referring act that may fail). Repair is usually chosen over replanning, unless a failure cannot be diagnosed, or a repair is deemed too expensive. This model addresses when repairs should be planned, and not the actual repairing of a referring expression, applying at a level below Heeman and Hirst's model.

Grosz, Sidner, and Lochbaum (Grosz and Sidner, 1990; Lochbaum, Grosz and Sidner, 1990) model how several agents with partial knowledge collaborate on constructing a *shared domain plan*. Each agent communicates their beliefs and intentions by making utterances about what actions they can contribute to the shared plan. Collaboration is again modelled by the agents establishing a mutual belief that each action in the shared plan contributes to the goal of the plan, and that each action can and will be performed by one of the agents. Unfortunately, an agent has no recourse, except to disagree, when it finds the shared plan invalid (i.e., a belief about an action in the plan is inconsistent with the agent's beliefs). Here, the agents intend to cooperate when they execute the domain plan, but not while negotiating who is going to do what.

In Grosz, Sidner, and Lochbaum's view, a shared plan contains only actions that are mutually believed to contribute to the plan. This view is too restrictive because an agent will sometimes propose an action that is not believed by the other agent, or even believed by himself to contribute. Without being able to incorporate these types of proposals into the shared plan, the model cannot represent the state in which an agent understands how an utterance contributes to the plan, while not agreeing with it. Heeman (1991) suggests that the only requirement for a shared plan is that it not be incoherent.

2.5 Implementation Details of Heeman's System

Since we are building upon Heeman's (1991) implementation, we describe the pertinent details here, thus avoiding a lengthy discussion later in the thesis.

The system is implemented in Prolog. The three central modules are the plan construction module, the plan inference module, and the belief module. We will describe each in turn, paraphrasing from (Heeman, 1991, pp. 86–92).

Plan Construction

The input to the plan constructor is an effect that the speaker wishes to achieve. The constructor then starts with a set of action schemas that satisfy the intended effect. Since the speaker's actual communicative goal cannot be achieved directly, he instead plans actions that achieve the goal indirectly. Thus, the effect that a speaker plans to achieve is to have the hearer recognize his communicative goal.

The output is an instantiated plan derivation that the speaker believes is valid. A plan derivation is an instance of an action schema that has each of the action schema headers in its decomposition recursively expanded into instances of action schemas. To be instantiated, a plan derivation must have all of its variables instantiated and unified. An agent views an instantiated plan derivation as valid if the agent believes that all of the constraints hold, and all of the mental actions are executable.

The plan construction algorithm is a best-first search. The *best-first heuristic* is to prefer plan derivations with the fewest number of primitive actions. The algorithm initializes a list of partial derivations to contain all of the instances of action schemas (from the initial set of action schemas) whose constraints are satisfiable. Each partial derivation in the list has an associated list of unexpanded steps that is initialized to be its decomposition. Until a plan derivation with no unexpanded steps is found, the algorithm repeatedly removes the partial derivation with the fewest number of primitive actions, finds all expansions of its leftmost unexpanded step, and adds these expansions (if any) to the list.

The expansion of a step depends on the type of action. To expand a mental action, the system performs the action immediately. Thus, a mental action can affect the rest of the construction. To expand a primitive action, the system instantiates the where-clauses,[9] ensures that the constraints are satisfiable, and ensures that the action is not already in the plan derivation. Finally, to expand an action schema header (that is not primitive), the system instantiates the where-clauses, ensures that the constraints are satisfiable, and adds its decomposition to the front of the list of unexpanded steps.

Plan Inference

Plan inference proceeds in two stages: plan recognition and plan evaluation.

The input to the plan recognizer is a set of observed surface speech actions (i.e., primitive actions). The plan recognizer outputs a set of plan derivations each of which is rooted at a specially marked top-level action schema and has only the input actions in its yield. It does not consider whether the constraints hold or whether the mental actions are executable.

The plan recognizer is based on chart parsing. It parses actions instead of words, so each edge in the chart represents a set of primitive actions accounted for by the edge. The strategy is essentially bottom-up, breadth-first. Since some action schemas have no observable primitive actions (e.g., those with a null decomposition), they would never be considered with a strictly bottom-up strategy, so an exception allows this type of schema to be added to the chart as an inactive edge, if necessary.

Each plan derivation in the set of recognized plan derivations is evaluated. If plan evaluation is successful, the output is an instantiated plan derivation that the hearer believes is valid. Otherwise, the evaluator outputs the constraint or mental action that caused the failure.

The algorithm proceeds as follows. It first instantiates all of the where-clauses. Second, it gathers all of the constraints and mental actions into a list. Third, it iteratively chooses the first unevaluated constraint or mental action subject to meta-knowledge that correctly orders the evaluation. For a constraint, it attempts to prove that it is satisfiable, and for a mental action, it attempts to execute it. If it fails in evaluating a constraint or mental action, it determines which constraint or mental action was the cause of the failure (which is not necessarily the constraint or mental action that could not be evaluated). Although the algorithm would prefer to evaluate the constraints and mental actions in the order in which they occur in the plan, this is not always possible because the action schemas have been formulated for plan construction. The meta-knowledge simply encodes which parameters of a predicate must be instantiated before it can be evaluated.

[9] If the where-clauses cannot be instantiated, the system aborts.

Belief Module

Heeman's belief module provides the minimum functionality to support plan construction, inference, and reasoning in a collaborative environment. Functions are provided for adding, testing, and retracting beliefs. Belief revision is not possible. Internally, a belief is encoded as `abel(N,Prop)` that represents the `Nth` alternating belief between the system and the user (Cohen and Levesque, 1990). Instead of using the `abel` predicate, the action schemas use the belief operators `bel(Agent,Prop)`, `mb(Agent1,Agent2,Prop)`, and `ab(Agent1,Agent2,Prop)`, representing simple belief, mutual belief, and alternating belief, respectively.

For reasoning about beliefs, he has taken the syntactic approach, with the addition of several inference rules. The rules can be applied within an arbitrary nesting of belief operators. One inference rule allows the system to believe that a proposition is mutually believed, if the system believes the proposition, and if the system believes that the user believes the proposition. A second inference rule allows the system to believe an alternating belief, if the system believes the proposition, or if the system believes the user has an alternating belief about it (applied to a maximum embedding of recursive beliefs) (Cohen and Levesque, 1990). A third rule, that attempts to capture co-presence, allows the system, for certain propositions, to believe that the user believes the proposition if the system believes the proposition.

Chapter 3

A Plan-Based Representation of Route Description

3.1 Introduction

Since we are considering the task of giving directions in *conversation*, we view direction-giving as communicative action. Therefore, we can use the planning paradigm to account for the way a set of directions is described or understood. At a high level, an agent, *the direction giver*, adopts a goal to have a second agent, *the recipient*, know how to get from A to B. For the recipient, this means knowing the route, that is, knowing the sequence of paths and places that link A to B (or, alternatively, the sequence of actions she can perform to get herself from A to B). To achieve his goal, the giver must both plan a route, and plan a description of that route—two tasks that are mostly independent but may be interleaved.

In a direction-giving dialogue, the giver refers to paths and places that the recipient has no knowledge of. The central focus of this thesis is to model the collaboration that occurs while making these references. However, before we can discuss reference itself, we must establish a framework, or a context, to motivate our theory of reference. This framework is a plan-based model of route descriptions, and we begin by discussing interactive direction-giving.

Interactive direction-giving presents us with some interesting phenomena to account for. First, in comparison to written directions (Riesbeck, 1980), which tend to be very complete by covering all possible contingencies (including detailed descriptions and error-correction information), conversation allows the directions to be less complete. Since the participants are able to collaborate, the direction giver need only make the directions good enough to some threshold he believes is adequate. If the recipient of the directions finds them inadequate (i.e., she believes that she will not be able to follow them), then she may simply ask for clarification or elaboration . This collaborative behaviour can be attributed to several sources, including time pressure (one doesn't want to spend all day giving directions), and the giver's ignorance of the recipient's knowledge (why guess, when one can assume) (Clark and Wilkes-Gibbs, 1986)[1].

Second, in comparison to the work in automated driving directions, in which directions are given dynamically and no complex driver intervention is allowed, directions given in conversation tend to be less specific. One major focus for giving automated driving directions is to instruct a driver in how to negotiate complex turns as the driver is making the turn. The specific details of the actions required of the driver simply cannot be conveyed completely by a linguistic description before the turn is negotiated (unless the driver has a photographic memory). In our task it is more important to describe the place a turn must be made, along with a summary of the expected difficulties a driver may have (such as getting into the left lane to make the turn). Again, because collaboration is possible, the recipient of the directions may query for additional information about the turn if they feel that it is necessary.

Two issues emerge out of the above discussion: determining the content of a route description, and collaborating on the description. Addressing the first issue, we note that there is a trade-off between making the directions simple enough to remember (or to write down quickly) and to be understood, while making them complete enough to be useful. How do people do this? In their route descriptions, the descriptions of locations where changes of direction are made are more important than the descriptions of the actions one performs at those locations (Psathas, 1991). Not only are the descriptions of locations important but they are kept relatively short by containing only the most salient features of the location (Davis, 1989).

We consider describing a route to be our domain task, even though the conventional view is that a domain task (for a task-oriented dialogue) should not be involved directly with communicative action (Litman and Allen, 1987; Grosz and Sidner, 1990; Lambert and Carberry, 1991; Heeman, 1993). Direction-giving in our context is a linguistic task, but it can be a sub-task of a higher domain task such as cooperatively planning a route.

The second issue, studied by Psathas, involves the collaborative aspects of direction-giving in conversation. People use many of the same actions that Clark and Wilkes-Gibbs discovered were used for collaborating on referring expressions. A route description can be judged and refashioned in much the same way as a referring expression is. Additionally, because the references to locations on the route are the most important part of a route description, collaboration is used to make sure that the references are adequate.

We limit the breadth of our investigation to the issues of collaboration on making an understanding route descriptions within the planning paradigm. The following assumptions

[1]Although Clark and Wilkes-Gibbs's principles of mutual responsibility and least collaborative effort are made in reference to referring expressions, they apply, in general, to collaborative dialogue.

are made about the content of route descriptions:

1. As is obvious from the previous discussion, we consider only *driving* directions. This implies a knowledge of Lynch's paths, nodes, and landmarks; specifically, roads, intersections, and features thereof (Lynch, 1960). We can also consider walking directions if we constrain them to only require spatial knowledge similar to that needed for driving directions.

2. The route description is to be remembered and executed at a later date. The direction recipient cannot execute the directions as she hears them. This assumption coincides with giving directions over the telephone (ignoring mobile phones), or on a street corner with a stranger.

3. We are concerned only with linguistic descriptions. Therefore, we do not account for deictic gestures, or references to physical maps.

4. The input to the model is a route in the form of an alternating sequence of direction actions (goto and turn) corresponding to the actions performed on paths or at nodes. This representation could be built easily from a sequence of links and nodes (see Frisch et al. (1990) and Cole et al. (1991)).

5. We take route descriptions out of context by not considering the conversation before or after the direction-giving dialogue. This amounts to assuming that the participants have the origin and destination in their common ground, along with the intention to describe/understand the directions. These issues are resolved in the opening and closing that Psathas (1991) describes.

6. Directions are usually broken up into *installments*, but we do not model this behaviour. Instead, a route description is given in a single utterance. However, this is an important topic, which we will consider in the discussion at the end of the chapter.

In this chapter we present a plan-based model of route description for which we have three requirements: a representation of spatial knowledge, action schemas for describing routes and locations, and processes for constructing and inferring route description plans. These requirements are presented in the next three sections, along with examples. The final section of the chapter is a discussion of some meta-issues of the model.

3.2 Representation of Spatial Knowledge

Direction-giving in a conversational context requires that the participants employ some spatial reasoning (Riesbeck, 1980). Therefore, in order to give or understand directions, we need an adequate representation of spatial knowledge. We adopt Kuipers's (1978) representation of the cognitive map, but we use only the features relevant to direction-giving: the topological structure and the route structure. While we do not use these structures as Kuipers does (to monitor action in an environment with the purpose of spatial reasoning), we do make use of them to describe and understand routes linguistically, something that his model does not take into account.

Kuipers's model uses metric information and topological information. We choose not to use the metric information directly in making descriptions, because people usually find the descriptions inadequate or unclear (Riesbeck, 1980). Topological information is one source of descriptiveness in directions, the *where* part, but we also need attributive information. Attributive information describes the *what* part (Downs and Stea, 1973). Kuipers does not consider the attributes of paths and places, which may include landmarks, street signs, street lights, etc. (see Davis (1989)). Attributive information is represented by a set of propositions that an agent believes are true of a path or place. These beliefs are accessed when constructing a referring expression for a path or a place. For example, an agent may believe that a place is a town, which has a name, or may believe that a path is a road with an asphalt surface.

In the next two sections we define the topological and route representations that we use. The topological structure is represented the same way that attributive information is, as a set of beliefs that are used when describing a route. The route structure is represented as a data structure provided as an argument to the action schemas defined in section 3.3. Following the definitions, is an example representation of a simple street network.

3.2.1 Topological Structure

The topological structure represents declarative knowledge about a street network by using two sub-structures: path descriptions and place descriptions. The path structure,

```
path( Name,
      Row ),
```

defines a path the unique internal name `Name`. The `Row` is a partial ordering (a list) of the places known to be on the path.

The place structure,

```
place( Name,
       Intersection,
       LocalGeometry ),
```

defines a place with the unique name `Name`. The `Intersection` field is a list of all paths that intersect at the place, that is, paths that pass through or end at the place. The local geometry describes, with metric information, how the intersecting paths relate to the place.

We do not use the local geometry, but we keep it for completeness and for possible future expansion.

Taken together, many instances of these structures represent a street network. Kuipers intended them to be able to incorporate new information inferred from traversing the network, or to give out information in answer to questions (i.e., the angle of a turn from one path to another at a place). In our model we will not use these properties of the network, but, for generation, we will use topological structure to help describe paths and places, and, for understanding, as a framework to be filled in as directions are understood. Knowledge of the topological structure will be stored as a set of propositions that an agent has beliefs about. Each path and place that an agent knows about is represented as a proposition that the agent believes is true.

3.2.2 Route Structure

The route structure is important in this model, because it contains the procedural knowledge of how to get from one place to another. A route could be considered as a series of paths and places (Frisch et al., 1990; Cole et al., 1991), but we consider it to be a sequence of actions (Kuipers, 1978) because a route represents action in the environment. A route description is an alternating sequence of the two possible actions that are taken as one traverses the street network. The two actions are to *go to a place*, and to *make a turn*. The first has the following representation:

```
goto( FromPlace,
      ToPlace,
      Path,
      Orientation,
      Distance )
```

It indicates that one travels from the `FromPlace` to the `ToPlace` along a `Path`. `Orientation` refers to a one-dimensional orientation (forward or backward) on the path, which is an internal quality that is not physically observable. It relates to the partial order of places, `Row`, contained in the path description. `Distance` is a metric quantity that we do not use directly. The `goto` structure represents an ordered pair of places between which no other actions are to be made, meaning that, to describe a path, we only need to consider the endpoints.

A second action is to make a turn, and has the following representation:

```
turn( Place,
      Path1,
      Orientation1,
      Angle,
      Path2,
      Orientation2 )
```

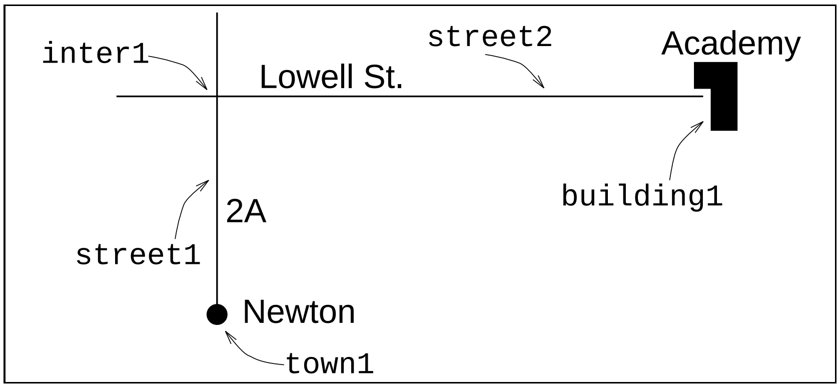

Figure 3.1: A street network

This indicates that in making the turn at `Place` one starts on `Path1` and ends up on `Path2`. The two orientation components are similar to those in the `goto` structure. The `Angle` is the angle of the turn made, a metric quantity that we do not use.

These two structures correspond to action schemas to be defined in section 3.3. Unlike the topological structures, they are not stored as beliefs. The set of actions (directions) for a particular route are grouped together to form a list that is an argument to the planning processes, either to be described (when giving directions), or filled in (when understanding a route description).

As we shall see in the next section, to sequentially plan the route description, we need to know how much of the route we have already described. To facilitate the process, we define a you-are-here structure (analogous to Kuipers's) that contains three aspects about a current position:

```
here( Place,
      Path,
      Orientation )
```

The planning processes maintain a current position pointer, represented as a `here` structure, and update it as a description is made or understood. The route origin and destination are also represented as `here` structures.

3.2.3 An Example Spatial Representation

Figure 3.1 presents a small street network. First, we list the attributive information in the form of propositions that an agent would have beliefs about.

```
category(town1,town).
called(town1,'Newton').
category(street1,street).
called(street1,'2A').
category(inter1,intersection).
```

```
category(street2,street).
called(street2,'Lowell').
category(building1,building).
called(building1,'Academy').
```

Briefly, `town1` is an instance of a `town` (the `category` proposition assigns the category of an object). The second proposition, `called`, assigns the name `Newton` to the town. The other definitions follow similarly.

The objects, `town1`, `inter1`, and `building1` are places, and have the following definitions within the topological structure:

```
place( town1,
       [street1],
       [(0,street1,+1)] ).

place( inter1,
       [street1, street2],
       [(0,street1,+1), (90,street2,+1),
        (180,street1,-1), (270,street2,-1)] ).

place( building1,
       [street2],
       [(270,street2,-1)] ).
```

In each structure, the first parameter is the internal name of the place. The second is a list of all paths that intersect at the place. For example, at `inter1`, `street1` and `street2` intersect. The third parameter expresses the local geometry as a list of triples comprising an angle, a path, and a one-dimensional orientation. Each triple represents the direction that the path leaves from the place (e.g., `street1` leaves to the north, with a forward orientation, and also to the south with a backward orientation). We include this information for completeness, but it can be safely ignored.

The second component of the topological structure represents the paths `street1` and `street2`:

```
path( street1,
      [town1, inter1] ).

path( street2,
      [inter1, building1] ).
```

These structures include the internal name of the path and a list of places on the path. The order of the places is important because it relates to the one-dimensional orientation seen in the place structures above.

We would like to describe a route going from Newton to the Academy traversing `town1`, `street1`, `inter1`, `street2`, and `building1`. Thus, we require the following route structures:

```
Origin:  here(town1, street1, +1)
Goto1:   goto(town1, inter1, street1, +1, 1)
Turn1:   turn(inter1, street1, +1, 90, street2, +1)
Goto2:   goto(inter1, building1, street2, +1, 1)
Dest:    here(building1, street2, +1)
```

The origin represents that we are at `town1`, facing forward on `street1`. The first action, `Goto1`, specifies that we go to `inter1` from `town1` by traversing `street1` in the forward direction, for a distance of 1 unit. The action to turn is similar, specifying that at `inter1` a turn (of 90 degrees) from `street1` onto `street2` is made. The actual route that needs to be described is the list of the three actions `Goto1`, `Turn1`, and `Goto2`, which is represented as the following route data structure that the planning processes operate with:

```
route([Goto1, Turn1, Goto2]).
```

The origin and destination are implicitly known to the giver and recipient because we assume that the agents involved have already agreed upon this information.

3.3 Route Description Action Schemas

We treat describing a route (from origin to destination) as an illocutionary speech act similar to informing. The goal of the act is to have the hearer understand the description, and hence know how to get from the origin to the destination by following a sequence of direction actions. The speaker attempts to achieve this goal by realizing the description as a series of utterances that correspond to surface speech acts that are primitive actions in the plan hierarchy.

Our route description action schemas are modelled very closely on Heeman and Hirst's refer action schemas (Heeman and Hirst, 1992). Plan decomposition is used to map the top-level action, through intermediate actions, into surface speech actions. The following is a list of the action schemas that we define along with their decompositions. The symbol $\stackrel{d}{\Longrightarrow}$ may be read as 'decomposes to'.

$$\begin{array}{ll} \texttt{describe-route} & \stackrel{d}{\Longrightarrow} \texttt{directions} \\ \texttt{directions} & \stackrel{d}{\Longrightarrow} \{\ \text{null} \mid \texttt{direction directions}\ \} \\ \texttt{direction} & \stackrel{d}{\Longrightarrow} \{\ \texttt{s-goto refer} \mid \texttt{s-turn refer}\ \} \end{array}$$

In the following sections we will call the physical actions (or their descriptions) that one performs while traversing a route "directions" in order not to confuse them with the communicative actions. In the schemas, we use the standard Prolog notation that variables begin with an upper-case letter and predicates and constants begin with lower-case.

Also note that all of the predicates and mental actions used in these schemas are defined in appendix A.

3.3.1 Top-level Action

Since describing a route is a communicative act, it has a top-level action schema called `describe-route` shown in figure 3.2. The describe-route schema takes as a parameter the

HEADER:	`describe-route(Route)`
WHERE:	`speaker(Speaker)`
	`hearer(Hearer)`
	`Route = route(Dirs)`
	`route-origin(Origin)`
	`route-dest(Dest)`
DECOMPOSITION:	`directions(Origin,Dirs,Dest)`
EFFECTS:	`bel(Hearer,goal(Speaker,knowroute(Hearer,Speaker,Route)))`

Figure 3.2: `describe-route` schema

route that consists of a list of directions, `Dirs`, which is instantiated by a WHERE-clause. Two WHERE-clauses instantiate the origin, `Origin`, and the destination, `Dest`. We assume that the origin and destination are known to the direction giver and recipient before the giver actually starts describing the route. Two WHERE-clauses also instantiate the `Speaker` and `Hearer` to `system` or `user` depending on whether plan construction or inference is being done.

The action decomposes into the single step `directions`, to be defined later in this section. The effect of `describe-route` is that the hearer will believe that the speaker has the goal of the hearer knowing the route, that is, knowing how to get from the origin to the destination. The effect is formulated this way because when a speaker has a communicative goal, he usually intends it to be recognized by the hearer (Searle, 1969; Appelt, 1985a; Grosz and Sidner, 1986; Heeman and Hirst, 1992). Whether or not the hearer believes that the goal (of her knowing the route) is achieved, she will know what the intended effect of the plan is through the plan inference process. The use of knowroute(A_1,A_2,R) is similar to Heeman's use of knowref(A_1,A_2,E) meaning that A_1 knows the referent that A_2 associates with the discourse entity `E`. For A_1 to know the route, `R`, that A_2 is describing entails A_1 inferring a complete list of directions `Dirs` that 'link' the `Origin` to the `Dest`.

Obviously, every schema should have one or more effects associated with it, but to simplify, only the top-level route description action and the surface speech actions have effects. Heeman makes the same simplification, stating that nothing is lost because we are chaining by decomposition, and plan inference always derives the correct plan derivation.[2] However, since the route description actions make use of referring actions, which have effects, the plan inference, constructing, and reasoning processes will have to take these effects into

[2] In Heeman's system, only the top-level `refer` action has an effect that captures all of the effects of the sub-actions in the plan.

account. In chapter 6, we describe the acceptance process that takes plans with multiple subplans and effects into account.

3.3.2 Intermediate Actions

The `directions` action is used to sequentially describe each of the directions in the list of directions in order to 'link' the origin to the destination. By using a current list of the directions that have not yet been described and a current position pointer (the you-are-here pointer) to coordinate its activity, the `directions` action determines the next direction to describe. Initially, the list contains all the directions needed to link the origin to the destination, and the current position is the origin; but, as the process continues, directions are pulled off from the head of the list, and described, until there are none left. The process of describing a route is sequential and iterative, but we model it recursively (tail recursion only) to make the planning simple.

Thus, we define two action schemas. The first, shown in figure 3.3, serves to terminate

HEADER:	`directions(CurPos,Dirs,Dest)`
CONSTRAINT:	`CurPos = Dest`
	`Dirs = [ ]`
DECOMPOSITION:	`null`

Figure 3.3: `directions` schema

the recursion. The schema has constraints that make sure we have reached the destination, and that all of the directions in the list have been accounted for.[3] The decomposition step is `null` in order to distinguish it from a primitive action.

The second schema, figure 3.4, uses its constraints to pull off the head of the direction

HEADER:	`directions(CurPos,Dirs,Dest)`
CONSTRAINT:	`Dirs = [Dir	RestDirs]`
DECOMPOSITION:	`direction(CurPos,Dir,NewPos)`
	`directions(NewPos,RestDirs,Dest)`

Figure 3.4: `directions` schema

list[4] which will be used to describe the next direction. This constraint also ensures that there is still further description to be done. The first step of the decomposition is to describe the next direction via the `direction` action that, as part of its decomposition, updates the current position pointer. The second step is recursive because it calls on the `directions` action with the new position pointer, and remaining directions as arguments.

[3] `[ ]` is Prolog notation for the empty list.

[4] `[H|T]` is Prolog notation for a list consisting of a head `H` (an element), and a tail `T` (a list).

The final intermediate action that we define is the `direction` action. There are two schemas to choose from, depending on the type of direction, corresponding to the directions of `goto` and `turn` defined in section 3.2.2. For both schemas the constraints make sure that the schema corresponds to the direction, and instantiate the necessary variables. The first schema, figure 3.5, is for describing movement along a path from one location to another. The first step of the decomposition associates the new location with its discourse entity,

```
HEADER:         direction(CurPos,Dir,NewPos)
CONSTRAINT:     CurPos = here(From,Path,Orient)
                Dir = goto(From,To,Path,Orient,Distance)
DECOMPOSITION:  ref(ToEntity,To)
                s-goto(ToEntity)
                set-context(From,To,Path,Orient)
                refer(ToEntity)
                NewPos = here(To,Path,Orient)
```

Figure 3.5: `direction` schema

which is used by the second step, the surface speech action `s-goto` (defined below). The third step is a mental action that sets the current context for making a reference to the location. The action takes into account all the locations along the path, from the current position to the new location, by consulting the topological structure. The current context is used by the fourth step that refers to the actual location. The `refer` action is described in detail in the next chapter. Finally, a mental action determines the new position.

The second schema, shown in figure 3.6, is similar to the first. In the decomposition,

```
HEADER:         direction(CurPos,Dir,NewPos)
CONSTRAINT:     CurPos = here(Place,Path1,Orient1)
                Dir = turn(Place,Path1,Orient1,Angle,Path2,Orient2)
DECOMPOSITION:  ref(PathEntity,Path2)
                s-turn(PathEntity)
                set-context(Place)
                refer(PathEntity)
                NewPos = here(Place,Path2,Orient2)
```

Figure 3.6: `direction` schema

a different surface speech action, `s-turn`, is used, and a reference to the path turned onto is made. The current context for the reference is computed by considering only the place at which the turn is being made. The mental action gathers all the paths and landmarks associated with the place by accessing the topological structure.

Note that these `direction` schemas fill the minimum requirements for giving directions: describing how to get to a place and how to make a turn. They use only some of the parameters that they have access to. For example, the schema for turning makes no reference to the angle[5] of the turn. Further action schemas (and surface speech actions) could be added easily to allow for more descriptive directions, but as we have observed, it is the descriptions of places at which changes in direction are made that take importance over the descriptions of the directions themselves.

3.3.3 Surface Speech Actions

The route descriptions must map into primitive surface speech actions that will be linguistically realized as parts of an utterance. We define two surface speech actions: `s-goto` for describing how to go to a place, and `s-turn` for describing how to make a turn. The action schema for `s-goto` is shown in figure 3.7. The constraint accesses the topological structure,

```
HEADER:        s-goto(PlaceEntity)
WHERE:         speaker(Speaker)
               hearer(Hearer)
               ref(PlaceEntity,Place)
CONSTRAINT:    place(Place,_,_)
EFFECTS:       bel(Hearer,goal(Speaker,
                          mb(Speaker,Hearer,place(Place,_,_))))
```

Figure 3.7: `s-goto` schema

and makes sure that the speaker believes that the place exists. This constraint cannot be evaluated unless the referring action that instantiates the entity (in the `direction` schema) is successful. The effect of the action is for the hearer to believe that it is mutually believed that `Place` is a place. As with the effect of the top-level action, this effect is formulated so that the intention behind the action is recognizable: the intention to establish mutual belief about the place.

The `s-goto` surface speech action corresponds to uttering a command for going to a place. It cannot stand alone, because it gets realized together with the associated referring expression for the place, which is realized as a result of the `refer` action in the `direction` action schema. So, along with a referring expression, this action could be realized with an utterance such as:

Go to the Lowell Street intersection.

There are many other ways to describe this particular direction, some giving more information than others. It would be possible to define more `s-goto` actions, which would take additional parameters (from information in the `direction` schema). For example, if an additional entity for the path of travel is given, one could utter the following, taken from Psathas's corpus (Psathas, 1991, p. 196):

[5] The numeric value of an angle can be converted to a descriptive phrase such as *right* or *sharp right* (Davis, 1989; Cole et al., 1991; McDermott, 1980).

Stay on 2A until ya get to Lowell Street.

Figure 3.8 shows the surface speech action for turning. Its constraint and effect are

HEADER:	`s-turn(PathEntity)`
WHERE:	`speaker(Speaker)`
	`hearer(Hearer)`
	`ref(PathEntity,Path)`
CONSTRAINT:	`path(Path,_)`
EFFECTS:	`bel(Hearer,goal(Speaker,mb(Speaker,Hearer,path(Path,_))))`

Figure 3.8: `s-turn` schema

similar to those of the `s-goto` action.

This action is used to describe a turn onto a path, so it must be realized together with a referring expression for the path. Again, this action is only one of many possible actions for turning—it could be realized as (with a referring expression):

Turn onto Lowell Street.

There are many other common ways of describing a turn, two of which are to utter:

Make a sharp right, or
Turn right at the intersection,

whose surface speech actions would require the turn angle and the place of the turn as parameters, respectively.

3.3.4 Referring to Paths and Places

What we have left out of the action schemas above is how a reference to a path or place is accomplished. The `refer` actions in the `direction` schemas decompose into referring expression plans, which have primitive actions for uttering the various components of the expression. In giving directions, the giver often makes references to paths and places that the recipient has no knowledge of. This type of reference is different from the type of reference (that, traditionally, has been studied) in which the two agents are copresent with, or have mutual knowledge of the referent. We hope that our plan-based model of route description has provided sufficient motivation for our model of reference, which follows in chapter 4.

3.4 Constructing and Inferring Route Description Plans

The system can act as both direction giver and recipient. As the giver, the system constructs route description plans, and as the recipient, the system infers route description plans.

3.4.1 Construction

We use the same plan construction process as Heeman does. To construct a route description for `Route`, the system gives the plan constructor the effect

```
bel(user,goal(system,knowroute(user,system,Route)))
```

as input. The constructor outputs an instantiated route description plan that the system believes is valid. The yield of the plan derivation is then output to the utterance generator (that we have not implemented).

An Example

Consider the scenario in which the system believes the spatial knowledge from the example above (section 3.2.3). We give the system, who has the role of speaker and direction-giver, the goal to describe the route from Newton to the Academy.

The system derives the plan shown in figure 3.9. In this and later plan derivation figures, the marked indentation pattern shows the decomposition. Constraints and effects are italicized. Variables have their first letter in upper-case, and co-referring variables have the same name. In this figure, `Origin`, `Dest`, `Goto1`, `Turn1`, and `Goto2` are the structures described in the above example. The plan derivation does not include the derivation of the referring actions, just the top-level `refer` action. The resulting surface speech acts of this plan are listed below (except for those related to referring).

```
s-goto(entity(1,inter1))
s-turn(entity(5,street2))
s-goto(entity(12,building1))
```

They correspond (with referring expressions added) to the utterances:

Go to the Lowell Street intersection,
turn onto Lowell Street,
go to the Academy building.

3.4.2 Inference

We use Heeman's plan inference process to understand a route description. Understanding involves recognizing the speaker's goals by examining his utterance. So, by inferring the speaker's plan, the hearer can determine the effects of the plan, which are formalized so that the hearer can easily recognize the speaker's intentions.

Plan inference proceeds in two stages. The first stage is to recognize a set of plan derivations that account for the observed surface speech actions, and the second stage is to evaluate each of the plan derivations in the set. Because of its top-level action, the hearer

```
describe-route([Goto1,Turn1,Goto2])
  directions(Origin,[Goto1,Turn1,Goto2],Dest)
    CONSTRAINT:  [Goto1,Turn1,Goto2] = [Goto1 | [Turn1,Goto2]]
    direction(Origin,Goto1,Dest)
      CONSTRAINT:  Origin = here(town1,street1,+1)
                   Goto1 = goto(town1,inter1,street1,+1,_)
      ref(entity(1,inter1),inter1)
      s-goto(entity(1,inter1))
        CONSTRAINT:  place(inter1,_,_)
        EFFECT:  bel(user,goal(system,mb(system,user,place(inter1,_,_))))
      set-context(town1,inter1,street1,+1)
      refer(entity(1,inter1))
      here(inter1,street1,+1)=here(inter1,street1,+1)
    directions(here(inter1,street1,+1),[Turn1,Goto2],Dest)
      CONSTRAINT:  [Turn1,Goto2] = [Turn1 | [Goto2]]
      direction(here(inter1,street1,+1),Turn1,here(inter1,street2,+1))
        CONSTRAINT:  here(inter1,street1,+1) = here(inter1,street1,+1)
                     Turn1 = turn(inter1,street1,+1,90,street2,+1)
        ref(entity(5,street2),street2)
        s-turn(entity(5,street2))
          CONSTRAINT:  path(street2,_)
          EFFECT:  bel(user,goal(system,mb(system,user,path(street2,_))))
        set-context(inter1)
        refer(entity(5,street2))
        here(inter1,street2,+1)=here(inter1,street2,+1)
      directions(here(inter1,street2,+1),[Goto2],Dest)
        CONSTRAINT:  [Goto2] = [Goto2 | [ ] ]
        direction(here(inter1,street2,+1),Goto2,Dest)
          CONSTRAINT:  here(inter1,street2,+1) = here(inter1,street2,+1)
                       Goto2 = goto(inter1,building2,street2,+1,_)
          ref(entity(12,building1),building1)
          set-context(inter1,building1,street2,+1)
          s-goto(entity(12,building1))
            CONSTRAINT:  place(building1,_,_)
            EFFECT:  bel(user,goal(system,mb(system,user,place(building1,_,_))))
          refer(entity(12,building1))
          here(building1,street2,+1)=here(building1,street2,+1)
        directions(here(building1,street2,+1),[ ],Dest)
          CONSTRAINT:  here(building1,street2,+1) = Dest)
                       [ ] = [ ]
          null
  EFFECT:  bel(user,goal(system,knowroute(user,system,[Goto1,Turn1,Goto2])))
```

Figure 3.9: Constructed route description plan derivation

will know that the inferred plan is a route description plan, thus she will believe that the speaker's intention is for her to know the route. If the hearer believes the plan is valid, then she knows the route, because all the constraints and mental actions are satisfiable. Otherwise, she believes that the route description is in error because either a constraint or mental action is found unsatisfiable. We discuss the different types of error in section 3.5.3.

For example, if the hearer observes the surface speech actions uttered by the speaker in the example above, then she will infer a route description plan almost identical to the one shown in figure 3.9. The only difference is that the discourse entities would be instantiated with different object names. We give a detailed example of the construction and understanding of route description plans in chapter 7.

3.5 Discussion

In this section we discuss a number of meta-issues about our model of direction-giving. First, we discuss how directions might be given in installments. Second, we compare direction-giving to referring. Third, we look at how a route description can break down, and fourth, we discuss the versatility of the model.

3.5.1 Giving Directions in Installments

In our model of route description, a description is uttered all at once, an implausible scenario given that route descriptions are usually long and might require repair. In dialogues between humans, directions are usually given in installments. That is, the direction giver breaks up his route description into a series of utterances, expecting the recipient to accept each one in turn. Humans use installments not only for direction-giving, but anytime they need to communicate complicated sequential information, such as a long address, or a long referring expression. Installments allow a person hearing the complicated information time to think about it and to, possibly, record it verbatim. It would be very difficult for one to understand a long set of directions or to collaborate on the repair of an inadequate portion of the directions, if they were uttered all at once. Another reason for a speaker to use installments is to receive confirmation or understanding for each part of a large plan before continuing to the next part. By using installments the speaker forces the hearer to consider the parts of a plan in a different order than they might have been considered had they been uttered all at once.

We have not modelled direction-giving in installments in this thesis because the topic is peripheral to the main focus—collaboration on referring expressions in direction-giving dialogues—and would needlessly complicate plan construction and inference, and the reasoning process. A second reason for not implementing installments is that we feel insufficient research has been done on the topic. Only Clark and Schaefer (Clark and Schaefer, 1989) touch on the subject in relation to making contributions to discourse.

At first glance the issue may seem simple to solve, but there are a few problems. Perhaps the most difficult problem is to determine the size of an installment. Intuitively, an installment should be the right size for easy comprehension. Should an installment be a single sentence, a speech act, a subplan of a larger plan, a plan with a maximum number of primitive actions, or a single direction action (in the direction-giving domain)? A second, but related problem, is to determine at what point installments are necessary. That is, what is the minimum size of plan that does not need to be uttered in installments? A third problem involves how installments are planned for, realized, and understood. Does an agent start planning a utterance, and then realize that installments are necessary, or does he plan to use installments ahead of time. By their nature, installments are incomplete plans, but they are not invalid because of this incompleteness.

Nevertheless, we suggest a way to model installments (for direction-giving) for which we do not yet have an implementation. First, the *size* of a plan is the number of primitive surface speech actions in it. We propose that the size of an installment depends on the number of surface speech actions in it. An installment would contain one or more complete direction actions, whose yield of surface speech actions would be near some maximum size allowance. Installments would not be used if the plan was short enough to be uttered in approximately one to two installments.

Second, how is an installment uttered and recognized? Clark and Schaefer discuss that humans place a rising or fall-rise intonation on all of the installments except the last, on which they place a falling intonation. This observation means that installments are marked. We propose that an agent plans a special surface speech action to indicate that his current utterance is an installment. For the last installment he would finish his plan as if it was a normal non-installment plan. The hearer can recognize the installment from the intonation, and can judge the validity of the installment separately, and incorporate it into the overall plan. Upon finding that the plan is incomplete and that the latest contribution was an installment, the agent can use a *continuer* (Clark and Wilkes-Gibbs, 1986; Clark and Schaefer, 1989), such as the utterance *yes*, or *okay*. A continuer expresses more than acknowledgement or acceptance. It also expresses that an agent believes the previous utterance was an installment, and is ready to hear the next installment. Thus, by using a continuer, an agent can postpone her judgement about the overall plan until later, and informs the other agent that she is doing so.

3.5.2 Comparing Direction-Giving to Referring

In one sense, the direction-giving task is analogous to the referring task, because one can think of describing a route as referring to the route. This is one reason that the action schemas for direction-giving are modelled closely after Heeman's schemas for referring expressions. There are, however, some fundamental differences between giving directions and referring to an object. Heeman uses a *candidate set* that represents the set of objects that the speaker believes could satisfy the propositional content of the current referring expression. The intermediate actions serve to narrow down the candidate set by adding further propositions to the referring expression, until the candidate set contains one object. So the process of referring involves a series of propositions that successively narrow down a candidate set. Describing a route could also be seen as narrowing down the set of all possible routes from A to B, by successively (recursively or iteratively) describing portions of the intended route. But, we see a route description as simply a list of actions forming a plan that, if performed, would get the performer from A to B. There is no candidate set to narrow down, only a list of actions (directions) to describe. This view leads to the three following observations.

First, the sequential structure of a set of directions is embedded in the route description plan (see Psathas (1991)). Thus, order is important within a route description plan because the order of directions is directly related to the order in which the directions must be performed. In referring expressions, there is no inherent order of modifiers, and their related propositions can be applied in any order to reduce the candidate set.[6]

The sequential ordering leads to the second observation: as each direction is given and mutually accepted, it can be added to the participants' common ground and not discussed again. This behaviour allows directions to be given in installments. For a referring expression, its modifiers tend to be kept active until the referring act is complete.

The third difference between Heeman's `refer` action and our route description action is that Heeman's action decomposition encodes what information is needed to describe a referent and how to communicate that information, whereas our actions are given the 'what' part as a list of directions, and the action decomposition only encodes how to communicate this list of directions. One purpose behind referring is to make a description that is minimal in size but maximal in descriptive capacity by choosing the best set of modifiers. Analogous to the modifier is the individual direction, and the best set of these is determined by a route planner that we have assumed.

So, although our direction-giving task is not equivalent to Heeman's referring task, we can use similar planning techniques to account for it.

[6] Although the order of modifiers within a referring expression may not be inherent, modifiers are usually ordered by their ability to maximally reduce the candidate set. That is, the most salient properties of an object are chosen and uttered first. Contrary to the order of directions in a route, the order of modifiers is context dependent.

3.5.3 Errors in Route Descriptions

When does a route description break down? That is, when does it not achieve its goal of describing a series of actions that can take an agent from one place to another? In Heeman's model, a referring plan is in error if the referring expression is overconstrained or underconstrained. These types of error do not occur with direction-giving, but an agent may find a route description, or a portion of it, in error for some other reasons.

One type of error occurs when an agent believes that the description of a path or a place is inadequate. Path and place descriptions are referring expressions and can be in error for the same reasons as regular referring expressions. But, since we are not assuming mutual knowledge of the paths and places, these referring expressions can also be *inadequate*. An expression cannot be overconstrained or underconstrained if one has no knowledge of the referent. Here, inadequate means that the agent lacks confidence in the whether the referring expression can be made into a plan good enough to be used to identify the referent when necessary. This type of error can be corrected by the agents use of a collaborative dialogue to elaborate on the referring expression. (See chapters 4 and 5 for a more complete discussion.)

A second type of error in a route description occurs if an agent believes that a direction is not descriptive enough. The direction itself could be too complex, or the description of the direction could be too simple. This type of error can be resolved by breaking the direction into sub-directions and planning descriptions for these, or by providing a more detailed description of the direction (by choosing a different `direction` schema that uses more of the information from the spatial structure). Since we have decided to concentrate on describing paths and places, we do not investigate this type of error in the thesis. This decision is reflected in the schema hierarchy by only having one `direction` schema for each type of direction, and by not modelling spatial reasoning about topological or route structures. In short, we can replan a description of a route, but not the route itself.

3.5.4 Versatility

Our plan-based model of direction-giving is versatile in a number of ways. First, additional action schemas can be designed easily to provide a wider range of descriptive ability. There is a linguistic variety (lexical and structural) of utterances that people use when giving directions, of which we only give a bare minimum: two schemas, one for going (on paths) and one for turning (at places).

Second, we use Kuipers's model of spatial knowledge representation, which includes a method of assimilating route knowledge into a topological structure. Thus, an agent who receives directions can incorporate them into her cognitive map. And an agent who already knows some geographical knowledge about a city can use this knowledge when discussing directions with the direction giver, possibly correcting mistakes, or adding directions herself.

Finally, because we use the planning paradigm to model linguistic direction-giving, we can incorporate non-linguistic actions into the plans. Actions such as pointing: *Go that way*; references to physical maps: *Follow the red line*; and other actions, such as nods of agreement, can all be added to make the model more comprehensive.

Chapter 4

Referring to Paths and Places

4.1 Introduction

From our previous discussion we have seen that the research to date into referring expressions has not dealt with what Appelt calls *nonshared concept activation with identification intention* (Appelt, 1985c). A speaker uses this action when he wants to refer to an object for which the speaker and hearer do not have mutual (or shared) knowledge with the underlying intention of having the hearer be able to eventually identify the object. This type of referring is prevalent in direction-giving dialogues because the direction recipient usually has no knowledge of at least part of the route (otherwise, why would she be getting directions?), whereas the direction giver has complete knowledge of the route. Because of this knowledge imbalance, the conventional models of referring are inadequate.

In this chapter we propose a method to generate and understand referring expressions without assuming the copresence of the agents and the referent, or the agents' mutual knowledge of the referent. In fact, the central assumption is that

(4.1) *the speaker and hearer have no mutual knowledge of any object that the speaker refers to.*

However, the speaker does have the intention to achieve a state of mutual knowledge about the object. The assumption is very strict, but we can make it in the direction-giving domain because most references are new (we are not considering anaphoric references).

The assumption is plausible because it leads to a different type of referring action that has a different underlying intention. Appelt's referring action is an attempt to formalize this action, and is distinct from his *shared concept activation* and Heeman's referring action (Heeman and Hirst, 1992). It is used when the speaker intends to refer to an object that he believes is not known by the hearer.

Consider a plan for a referring expression. What constitutes grounds for accepting the plan? In Heeman's system, the plan is achieved (and is, therefore, acceptable) if it uniquely identifies an object within the world of objects that an agent knows about. But how can an agent identify the referent without knowledge of it? We suggest that a referring expression plan can be accepted if

1. the plan contains a description that is *useful* for making an *identification plan* that the hearer can execute to identify the referent, and
2. the hearer is *confident* that the identification plan is *adequate*.

The first condition, first described by Appelt, is important because the referring action's success depends on the hearer formulating a useful identification plan. This formulation is not always possible—if a person on a bus asks *what stop should I get off at*, the response *one stop before I do* is not very useful (Appelt, 1985c, p. 202)—so the speaker must endeavour to use an appropriate description.

For the second condition to hold, the hearer must believe that the identification plan is good enough to uniquely identify the referent when it becomes visible. This involves giving enough information by using the most visually prominent or *salient* features of the referent.

The scope of the task is very broad, so we only consider some of the issues. We sidestep the issue of creating useful identification plans from referring expressions by assuming that the propositional content of any referring expression is always useful for making an identification plan.[1] We do not model how an agent can use the propositional content to actually identify the referent. This process occurs after the dialogue about identifying the referent, our chief concern, is over.

In this chapter we first discuss, in section 4.2, how an agent can judge her confidence in the adequacy of a referring expression by considering the salience of the referent's descriptors. This section also looks at the various uses of salience in our model. Then, in section 4.3, we discuss how an agent can apply confidence and salience to the construction and inference of plans. Section 4.4 defines the referring action schemas, and section 4.5 presents an example of generating and understanding a referring plan. The final section is a summary of the chapter.

4.2 Confidence in a Referring Expression

For both construction and inference, we suggest that evaluating one's confidence in a referring expression plan is the means of determining whether or not the plan is an adequate identification plan. By *confidence*, we mean an agent's belief that the plan is good enough, or adequate, to uniquely identify the referent once it becomes possible to do so (i.e., the agent must believe that she will be able to recognize the referent, eventually).

[1] In fact, we take the referring expression plan itself to be the identification plan, a view slightly different from Appelt's. His identification plans enable the hearer to *locate* and *identify* the referent, whereas ours only enable the hearer to identify the referent once the hearer establishes the relevant context (and, by following the directions, the hearer should have the correct context).

In the domain of direction-giving, the direction giver must be satisfied that the referring expression that he constructs will be helpful to the recipient in identifying the referent. To do so, he evaluates his confidence in the plan's adequacy cumulatively as he constructs the plan. When he is confident enough, he utters the resulting expression.

Upon hearing a referring expression, the direction recipient infers the associated plan, and evaluates her confidence in its adequacy. If she is confident that the plan will be useful enough, the direction giver can continue with the rest of the directions; otherwise she must initiate a collaborative dialogue that will serve to make her confident in its adequacy.

In this section we discuss how confidence is evaluated, and follow that with a discussion of salience on which confidence is based.

4.2.1 Confidence

Confidence in a plan's adequacy comes from two sources: quantity and quality. If a referring expression contains a long list of a variety of descriptors, then one might be confident that it is adequate. Alternatively, it might contain a few very *salient* features of the referent and thus be adequate. We propose that the old proverb of quality over quantity applies here because people usually try to minimize the length of their referring expressions (Reiter, 1990; Reiter and Dale, 1992; Heeman and Hirst, 1992), or because they are forced to make short expressions because of time pressure (Clark and Wilkes-Gibbs, 1986).

To evaluate the adequacy of a referring expression plan, we associate a numeric *confidence value* with the plan. Even though this value is numeric, it can be interpreted as ranging from low confidence to high confidence. An agent is confident in a plan if the confidence value of the plan exceeds some threshold. Ideally, the *confidence threshold* should vary depending on the context surrounding the utterance, but since we are only considering referring expressions uttered within one context (part of a direction instruction), the threshold may remain constant.

We also associate confidence values with each of the descriptors of a referring expression (i.e., the head noun and its attributes). These values can be thought of as the amount confidence is increased because of the use of a descriptor. The overall confidence value is composed of the confidence values of its descriptors. Since the values are numeric we simply add them up. One could envision more complex systems to evaluate confidence including an algebra of confidence (the interaction of descriptors may lower or raise confidence), or a non-numeric system, but we feel that this system is simple and does the job. The problem is that it is open to the ad hoc assignment of confidence values, an issue we discuss in the next section about salience.

Another point to consider is how sub-expressions influence the confidence of a main referring expression. An agent uses a sub-expression when he wants to relate the referent to another object (for example, in *the creature on the television*, *the television* is a sub-expression of *the creature*). Because the sub-expression forms a complete reference to an object, it has a confidence value associated with it, but how should this value affect the value of the main expression? We choose to ignore the secondary value since its associated referring expression is far less important than the main expression.[2] However, we do associate a confidence value with the type of relation between the referent and secondary object, which is added to the overall confidence value.

All confidence calculations are incorporated into the referring action schemas (to be defined in section 4.4) as mental actions. This means that evaluating the confidence of a referring expression plan falls out of the regular plan evaluation process. Plan evaluation also checks if the confidence value exceeds the threshold. Once a plan has been evaluated, an agent adds beliefs about it to her mental state. If the plan is valid, the agent adds a belief that the referring plan is adequate and successful; otherwise, she considers the plan inadequate and adds a belief that it is in error.[3]

4.2.2 Salience

We have discussed how confidence values are manipulated and evaluated, but where do they come from? We propose that the confidence value of a property is equivalent to its salience within the context of the referring expression.

Salience plays a key role in selecting landmarks for describing a location (Pattabhiraman and Cercone, 1990). One can think of many ways to compute the salient properties of a given situation. Davis (1989) uses a hierarchy of salient landmarks to determine which landmarks in a given location should be used to describe the location.[4] The most salient ones, such as traffic lights and buildings, are chosen first over other landmarks such as street names. Obviously a hierarchy should depend on the context of the situation. Lynch (1960) identifies the point-of-view, familiarity, and current goals of an agent as important contextual factors, and Devlin (1976) says salience may be influenced by identifiability, visibility, prominence, and functional importance.

For example, different landmarks may be salient to different individuals (consider street signs in Greece from the point of view of an English speaker). Furthermore, what is salient in one city might not be salient in another (consider the long, straight streets of Toronto, compared to those of London, which tend to meander and be re-named). Despite these considerations, Davis's hierarchy is a hard-coded list of what is salient.

[2] Alternatively, the value could be scaled down so that it contributes to the overall confidence a little bit, but this would involve formalizing an algebra of confidence—what about sub-sub-expressions?

[3] We should also note that sub-expressions of a main referring expression are evaluated in the same way except that they are always found adequate (i.e., an agent is always confident in a sub-expression). This view has the important ramification that a sub-expression will never be elaborated on by the collaborative model (described in chapter 6). Only the main level plan can be elaborated on. We made this decision to keep the dialogue simple by avoiding complex deep elaborations.

[4] The hierarchy is supposed to represent what the speaker believes the hearer believes is salient.

Reiter and Dale (1992) use a similar hierarchy of features to determine what features should be used first when describing a object. They say it is important to use the most visually prominent features of an object, such as size, colour, and shape. Again, their hierarchy is static because it does not vary depending on the context or on the type of object being described.

Ideally, as mentioned above, salience should depend on the context of the situation. For example, a tall building, which would normally be salient, would not be if it were surrounded by other tall buildings. The salience of a referent or of its properties should be calculated by taking the other objects in the context of the referent into consideration. This calculation would be quite complex and is beyond the scope of this thesis. But we would like to find a middle ground between the simple context-independent approaches of Davis, and Reiter and Dale, and a full blown contextual analysis.

The middle ground involves taking the type of object into account when choosing properties and landmarks that relate to it. This method is still simple enough to be encoded as a hierarchy but is more dependent on the context. For example, height can be a very salient feature when describing a building, but not when describing a street or a street sign. Similarly, having a sign is an important feature for an intersection, but not so important for a building (for which height, colour, and architectural style probably take precedence).

We would also like our model of salience to be useful for both the direction giver and recipient. There is a difference in their tasks which is reflected in how they make use of salient information. The direction giver knows the situation, or the context, and can compute what he believes to be salient from that situation directly. On the other hand, the recipient has no knowledge of the situation and can only make access to what she believes is generally salient. In our model, we formalize only the latter type of salience within two hierarchies. The recipient can make direct access to the hierarchies to determine confidence values and salient feature types. The giver can also access these hierarchies and couple them with his beliefs about the objects in the world to construct salient references to objects.

Salience is based primarily on visual salience, but also on what properties people find important in referring expressions (in the direction-giving domain). We require two salience hierarchies that encode this knowledge. The first is for object types (or categories), and the second is for object properties.

The *category salience hierarchy* is defined to be a partial ordering on the set of object categories. This is a simple hierarchy very similar to what Davis proposed. For us, it links the category of an object to how confident one can be when using this object as a referent. The following is a sample of a category salience hierarchy that an agent might have:

```
salient-category(4,town).
salient-category(3,street).
salient-category(1,intersection).
```

```
salient-category(1,sign).
salient-category(1,building).
```

In this hierarchy, towns are the most salient, followed by streets, intersections, signs, and buildings. The numeric values are equivalent to the confidence values for using the various types of objects.

The second salience hierarchy, the *property salience hierarchy*, involves the properties of objects. Since the salience of an object's property depends on the object, the hierarchy is actually a set of partial orderings of features indexed by object type. The hierarchy encodes how confident one can be by using each feature, represented as a lambda expression. For simple features, such as height, the object type is specified, while, for properties that relate one object to another, both objects' types are specified. Here is a sample property salience hierarchy:

```
salient-property(1, town,               λX·called(X,Name)).
salient-property(3, street,sign         λX·λY·has(X,Y)).
salient-property(2, street,             λX·called(X,Name)).
salient-property(3, intersection,sign,  λX·λY·has(X,Y)).
salient-property(2, intersection,       λX·called(X,Name)).
salient-property(4, building,           λX·height(X,Height)).
salient-property(3, building,           λX·colour(X,Colour)).
```

The most salient property of a town is its name. For streets and intersections, the most salient property is having a sign followed by having a name. For buildings, height is very salient followed by its colour. The numeric values are equated to the confidence values of each of the features.

In our model, the salience hierarchies are represented as beliefs. In effect, agents have beliefs about the salience of object categories and features. But when constructing a referring expression, a speaker should use features that depend on more than his own beliefs about salience. He should use features that he believes the hearer believes are salient. On the other hand, the hearer cannot know what will be salient in the context of the referent because she has no knowledge of its location. Perhaps, while describing the referent, the speaker also makes it clear that the features he is using are salient. The interplay of salience and reference has not been researched sufficiently in the literature and is a difficult issue to resolve; so we assume that the participants have a mutual knowledge of what is salient, in general, thus approximating the recursive nature of beliefs about salience. We leave a more complex model to future research.

The salience hierarchies are used for two purposes. The first is for generating salient referring expressions by choosing the most salient properties of an object first. In this capacity, they are used for constructing initial referring expressions, elaborating on inadequate referring expressions, and for suggesting possible elaborations to referring expressions (the latter two uses are discussed in the next chapter, and the former in the following sections).

The second role of the hierarchies is for evaluating one's confidence in the adequacy of a plan. Since the hierarchies contain confidence values, the evaluation process accesses these to calculate the overall confidence value of a plan.

Finally, note that choosing a salience hierarchy is a difficult task which undoubtedly requires much psycholinguistic and sociological research. We do not consider it further in this thesis.

4.3 Construction and Inference

In the previous section we looked at how confidence in the adequacy of a referring expression is assessed, and how salience is used for this purpose. In this section we describe how confidence and salience are applied to the construction an inference of referring expression plans.

4.3.1 Plan Construction

To refer to an object for which the speaker and hearer have no mutual knowledge, the speaker adopts the goal of informing the hearer about how to identify the object. This goal can be achieved by a referring expression plan that *uniquely specifies* the referent according to the speaker's beliefs and for which the speaker is *confident*. To this end, the speaker has two sources of descriptive power: his beliefs about the properties of the referent, and his beliefs about which properties of the referent are the most salient.

The propositional content of a referring plan is constructed by successively choosing properties that reduce a *candidate set*, the set of objects that satisfy the already constructed propositional content. When the candidate set contains one element, the referent, the plan uniquely specifies the referent according to the speaker's beliefs. But the goal of the referring action is not achieved unless the speaker is confident in the adequacy of the expression he is building. So, at the same time as the propositional content is being built, the overall confidence value of the plan is computed. The confidence values for each property are cumulatively summed up. Plan construction can finally terminate when the confidence value exceeds the confidence threshold (and the candidate set contains one element).

This, however, is not the whole story. The speaker also wants to make the most salient referring expression possible, so he does not choose properties indiscriminately. Instead, he chooses the most salient properties for the category of object he is referring to by accessing his beliefs about his property salience hierarchy. In so doing, he creates a referring expression that is inherently short (not necessarily minimal) with respect to the number of features because, intuitively, salience is linked to descriptive power. This result corresponds to the psychological evidence cited by Reiter and Dale (1992) that suggests humans make short descriptions not by minimizing the length of the expression, but simply by using salient information.

This construction process is then very similar to Heeman's planning process. The plan constructor has dominion over structural decisions, such as the order in which competing plan schemas are added to the plan being constructed. In fact, to make minimal plans, Heeman's plan constructor uses a *best-first heuristic*, which chooses plan schemas with the least number of primitive actions first. We still use this heuristic, but the plan schemas themselves also influence the size of a plan, because they determine which modifier should be added next.[5] In Heeman's model, modifiers are chosen non-deterministically, a choice influenced only by the planning heuristic.

This major difference from Heeman's model is encoded in the modifier plan schemas by using mental actions that select the most salient property of the referent that has not yet been used. Paired with this mental action is a second which accesses the salience hierarchy to determine the confidence value for using that modifier. These schemas are described in detail in section 4.4.2 below.

We have also modified Heeman's model to account for assumption (4.1). In order to use a property as a descriptor, we no longer require that the speaker believe that it is mutually believed, but only that he believes it to be true.[6] This change is made in the constraints of the primitive surface speech actions described in section 4.4.3 below.

The speaker's goal (for the hearer to be able to identify the referent) is encoded as an effect of the referring plan, specifically of the top-level `refer` action (see section 4.4.1 below). Additionally, the speaker has the intention to establish the mutual belief of all the properties that he uses. Therefore, the primitive actions have the effect of the making the properties mutually believed. Thus, after a plan is mutually accepted, the speaker and hearer have established mutual knowledge of the properties of the referent contained in the propositional content of the referring plan.

4.3.2 Plan Inference

The first step in plan inference is to derive all the plans that account for the observed primitive actions (surface speech actions). The second is to evaluate these plans. To evaluate a plan in Heeman's system, the evaluator attempts to instantiate the variables such that all the constraints hold and all the mental actions are satisfiable. The referring plan cannot be evaluated if it does not uniquely identify the referent according to the mutual knowledge of the speaker and hearer. In this case the hearer finds the plan either overconstrained or underconstrained.

[5]Although the plan schemas determine the most salient modifier that should be added next, there is still a non-deterministic choice between the two types of modifier: simple modifiers, and modifiers that relate the referent to an object. (see section 4.4)

[6]We remove Heeman's inference rule that states the hearer can infer mutual belief of a property if she believes it. This rule was intended to capture co-presence, which we have eliminated.

We will use this same evaluation process, but, as it stands, the hearer would always find one of our referring plans overconstrained. This error happens because the hearer has no knowledge whatsoever of the intended referent.

From assumption (4.1), the hearer knows nothing about any objects that are referred to, and, clearly, the speaker intends to refer to a new entity.[7] Therefore, the first modification we make to Heeman's plan evaluator is to initially scan the referring plan for each entity referred to. For the hearer, each entity is new, so she updates her belief about the objects in the current context, or focus of attention (Grosz and Sidner, 1986).[8] We create a new object name for each entity referred to, and add these to the hearer's current context set. The hearer doesn't have any beliefs about the properties of the new objects yet, just the belief that they exist. In other words, the hearer believes that some new objects exist, to which she can attach the corresponding properties in the referring plan.

The evaluation process is still inadequate, because the hearer has no beliefs about the properties of the referent that are used in the plan. The constraints in the primitive actions corresponding with these properties cannot hold. Again, from assumption (4.1), we make our second modification. When the hearer evaluates a constraint corresponding to a property of the referent, she infers that the speaker believes the property is true, and so she can add, to her own beliefs, the belief that speaker believes this property. However, she makes no assessment of her belief in the property.

With these two modifications in place, the hearer can now evaluate a referring plan without having any knowledge of the intended referent. But how is her knowledge of the referent and thus mutual knowledge of it established? Recall that the effects of the primitive actions of a referring plan are to establish the mutual belief of the referent's properties. If we assume that the hearer believes the speaker is rational, that is, sincere and competent (Cohen and Levesque, 1985; Appelt and Kronfeld, 1987), then the hearer can infer her own belief about a property of the referent if she believes that the speaker believes the property (and she believes this as a result of evaluating the plan). Then, mutual belief of the property is established once she accepts the referring plan and communicates her acceptance to the speaker. (See chapter 6, where we described the acceptance process.)

As in Heeman's system, plan evaluation fails if a constraint does not hold, or if a mental action is not satisfiable.

The main failure that we consider is when the confidence threshold is not exceeded, implying that the hearer is not confident in the adequacy of the plan. Since we use the same plan schemas for both construction and inference, confidence value computation and evaluation falls out of plan inference. A constraint makes sure that the overall confidence

[7]The hearer recognizes this intention as a result of deriving the plan.

[8]Each agent maintains a belief about what objects are in focus in the current context. The current context is simply a set of objects.

value exceeds the confidence threshold, and if not, an error is signalled.

A second type of failure occurs when the plan does not uniquely specify the referent. This failure can only be detected by the direction giver, since he has a set of objects that he must distinguish the referent from. For the direction recipient, any reference is new and so must be unique.

A third type of failure would happen at the constraints that correspond to the properties of an object, but, because the hearer does not believe these properties yet, we allow them to always hold because they will eventually be made true when the effects of the plan occur.

Therefore, if the confidence threshold is exceeded, evaluation is usually successful because the possibilities that lead to other failures are not common. They occur only if the plan is ill-formed, a condition we can detect, but have no recourse to repair.

4.4 Referring Action Schemas

As with the route description schemas, we draw on Heeman's referring schemas once again to formulate our new referring action schemas. These schemas are very similar to Heeman's because we want to keep the structure of the referring expression plan derivation the same. The two main differences, as discussed in the previous sections, are the constraints and effects of the primitive surface speech actions, and the incorporation of mental actions for evaluating confidence. The plan decomposition is the same as Heeman's:

$$
\begin{array}{lll}
\texttt{refer} & \stackrel{d}{\Longrightarrow} & \texttt{s-refer describe} \\
\texttt{describe} & \stackrel{d}{\Longrightarrow} & \texttt{headnoun modifiers} \\
\texttt{headnoun} & \stackrel{d}{\Longrightarrow} & \texttt{s-attrib} \\
\texttt{modifiers} & \stackrel{d}{\Longrightarrow} & \{ \text{ null} \mid \texttt{modifier modifiers} \} \\
\texttt{modifier} & \stackrel{d}{\Longrightarrow} & \{ \texttt{s-attrib} \mid \texttt{s-attrib-rel refer} \}
\end{array}
$$

We first show the top-level refer action, followed by the intermediate actions, and the surface speech actions.

The predicates and mental actions used in these schemas are defined in appendix A.

4.4.1 Refer Action

The top-level action, called `refer`, is shown in figure 4.1. As with Heeman's `refer` action, this action decomposes into two steps, one to communicate the intention to refer (`s-refer`), and the second to `describe` the referent. The header has two parameters: the entity that is referred to by the plan, and the overall confidence value of the plan that is computed within the plan.

The effect of this action is that the hearer believes that the speaker has the goal of the hearer identifying the referent by using the description. The effect is formulated in this way

HEADER:	`refer(Entity,OvrConf)`
WHERE:	`speaker(Speaker)`
	`hearer(Hearer)`
DECOMPOSITION:	`s-refer(Entity)`
	`describe(Entity,OvrConf)`
EFFECTS:	`bel(Hearer,goal(Speaker,identify(Hearer,Speaker,Entity)))`

Figure 4.1: `refer` schema

so that the hearer can recognize the speaker's intention, without having to believe that the goal is achieved. The goal `identify`(A_1,A_2,E) is used to distinguish this referring action from Heeman's, which has the goal `knowref`(A_1,A_2,E). The goal of our referring action is not for the hearer to know the referent immediately according to mutual knowledge, but to know how to identify the referent using the propositional content of the plan. A hearer of our referring act will actually know the referent, a perlocutionary effect, once she is able to find the referent in the environment. She believes that the `identify` goal is achieved if she believes that she is confident in the plan's adequacy as an identification plan.

4.4.2 Intermediate Actions

The `describe` action schema (figure 4.2) is used to construct a description of the referent by first choosing the head noun, and then by choosing the modifiers. In the decomposition,

HEADER:	`describe(Entity,OvrConf)`
DECOMPOSITION:	`context-current(Context)`
	`headnoun(Context,Entity,Cand,UsedPred,HeadConf)`
	`modifiers(Entity,Cand,UsedPred,HeadConf,OvrConf)`

Figure 4.2: `describe` schema

the current context, `Context`, is determined by a mental action. `Context` is a set of objects which the speaker has in his focus of attention. It is all those objects from which the referent must be disambiguated and all those objects which may be helpful for describing the referent.[9]

The `headnoun` schema returns the initial candidate set (all objects of the same category as the referent), the actual predicate used, `UsedPred`, and the confidence value of using the particular object type. These three values are supplied to the `modifiers` plan schema, which constructs the actual description.

[9] In his model, Heeman took referring out of context by modelling the world as a set of objects that an agent believed to exist. Since our referring action is made in many different situations over the course of a direction-giving dialogue, we have overlayed the world with a simple model of attention (Grosz and Sidner, 1986) that specifies which objects from the world are currently being attended to.

To recapitulate the planning process, we describe the use of these three values. The candidate set enables the plan constructor to disambiguate the referent from the other objects in the current situation. It is successively pruned down as new modifiers are added to the plan. The set of used predicates is a set of instantiated predicates that have already been used in the plan; this enables the constructor to choose salient properties that have not already been used. Finally, the confidence value is used to initialize the overall confidence sum that is cumulatively computed, and checked against the confidence threshold.

Figure 4.3 shows the `headnoun` schema, whose first step is to communicate the head noun of the description (via the `s-attrib` surface speech action). Second, the initial candidate set is computed by finding the subset of objects in the current context, `Context`, that are members of `Category`. Third, the object's internal name, `Object`, is determined, and is used

```
HEADER:         headnoun(Context,Entity,Cand,Pred,Conf)
WHERE:          speaker(Speaker)
                hearer(Hearer)
DECOMPOSITION:  s-attrib(Entity,λX·category(X,Category))
                subset(Context,
                       λX·ab(Speaker,Hearer,category(X,Category)),Cand)
                ref(Entity,Object)
                Pred = [category(Object,Category)]
                confidence-headnoun(Conf,Entity,Category)
```

Figure 4.3: `headnoun` schema

in the fourth step to initialize the set of used predicates. And finally, a mental action returns the confidence value of the head noun by accessing the agent's category salience hierarchy. The value represents how salient a reference to an object of this particular category is.

The `modifiers` schema is a recursive plan that enables the speaker to construct a salient description that uniquely determines the referent with respect to the speaker's beliefs, and for which the confidence value exceeds the threshold. It is formulated with two schemas. The first, shown in figure 4.4, embodies the recursion. It has a step to construct one

```
HEADER:         modifiers(Entity,Cand,UsedPred,CurConf,OvrConf)
DECOMPOSITION:  modifier(Entity,Cand,NewCand,UsedPred,NewUsedPred,ModConf)
                confidence-add(NewConf,CurConf,ModConf)
                modifiers(Entity,NewCand,NewUsedPred,NewConf,OvrConf)
```

Figure 4.4: `modifiers` schema

modifier, a mental action to add the confidence value of the modifier to the total, and a step to recursively call itself. The recursive use of this action accomplishes several things. First, the candidate set is reduced by each `modifier` step while the used-predicate set is expanded. And second, the overall confidence value is computed cumulatively by each `confidence-add` action.

The second `modifiers` schema (figure 4.5) terminates the recursion. The first constraint

HEADER:	`modifiers(Entity,Cand,UsedPred,Conf,Conf)`
WHERE:	`ref(Entity,Object)`
CONSTRAINTS:	`Cand = [Object]`
	`confidence-exceed(Entity,Conf)`
DECOMPOSITION:	`null`

Figure 4.5: `modifiers` schema

makes sure the candidate set only contains one object (that is the referent). The second constraint makes sure that the confidence threshold is exceeded by the confidence value, which becomes the overall confidence value of the plan (unified in the header). The decomposition is `null` to distinguish this action from the primitive surface speech actions.[10]

The `modifier` actions are used to construct modifiers for the head noun that prune down the candidate set by choosing the most salient property of the referent possible. Two examples are shown in figures 4.6 and 4.7. The first constructs a modifier for a simple

HEADER:	`modifier(Entity,Cand,NewCand,UsedPred,NewUsedPred,Conf)`
WHERE:	`speaker(Speaker)`
	`hearer(Hearer)`
DECOMPOSITION:	`salient-attrib(Entity,Pred,UsedPred,NewUsedPred)`
	`s-attrib(Entity,Pred)`
	`subset(Cand,λX·ab(Speaker,Hearer,Pred(X)),NewCand)`
	`confidence-attrib(Conf,Entity,Pred)`

Figure 4.6: `modifier` schema

attribute and the second describes the referent relative to another object. Both have a mental action that computes the most salient predicate to use that has not already been used by consulting the speaker's property salience hierarchy. As a side-effect the actions update the used-predicate set.[11]

Both actions have a surface speech action as a step in their decomposition, and both calculate how confident an agent can be by using the modifier. The mental actions, `confidence-attrib` and `confidence-attrib-rel`, access the agent's property salience hierarchy. In the second schema, we ignore the confidence value, `OtherConf`, of the reference to the other object because we have no theory to account for how this value could be used

[10]Notice that, while the candidate set, the used predicate set, and the cumulative confidence sum are modified and passed *down* the plan derivation, the overall confidence value is passed *up* the plan derivation, so that it ends up in the top-level `refer` action. In comparison, the other values end up in the last `modifiers` action.

[11]When evaluating a plan, these mental actions simply update the used-predicate set. They have no need to find a salient property because that property is supplied by the `Pred` parameter of the surface speech act.

```
HEADER:         modifier(Entity,Cand,NewCand,UsedPred,NewUsedPred,Conf)
WHERE:          speaker(Speaker)
                hearer(Hearer)
DECOMPOSITION:  salient-attrib-rel(Entity,OtherEntity,
                                   Pred,UsedPred,NewUsedPred)
                s-attrib-rel(Entity,OtherEntity,Pred)
                ref(OtherEntity,Other)
                subset(Cand,λX·ab(Speaker,Hearer,Pred(X)(Other)),NewCand)
                refer(OtherEntity,OtherConf)
                confidence-attrib-rel(Conf,Entity,OtherEntity,Pred)
```

Figure 4.7: `modifier` schema

(see section 4.2.1). However, the addition of another mental action to the schema would be simple once such a theory is developed.

4.4.3 Surface Speech Actions

For referring expressions we define three types of surface speech action. The first, shown in figure 4.8, is used by the speaker to inform the hearer that he is about to refer to an object.

```
HEADER:         s-refer(Entity)
```

Figure 4.8: `s-refer` schema

It is a distinct action because it is sometimes linguistically marked by a definite article, or by a phrase such as *See the dog?*

The second type of surface speech action, `s-attrib`, expresses an attribute of an object. Two examples are shown (figures 4.9 and 4.10) corresponding to uttering the category of an object, and the 'name' of an object. These actions are fundamental to constructing

```
HEADER:         s-attrib(Entity,λX·category(X,Category))
WHERE:          speaker(Speaker)
                hearer(Hearer)
                ref(Entity,Object)
CONSTRAINTS:    category(Object,Category)
EFFECTS:        bel(Hearer,goal(Speaker,mb(Speaker,Hearer,
                                     category(Object,Category))))
```

Figure 4.9: `s-attrib` schema (for `category`)

referring expression plans under assumption (4.1). To use these actions the speaker is constrained to believe that the attribute is true. The speaker's intention that the speaker and hearer establish a mutual belief about the attribute is encoded as an effect, and is thus a side-effect of the referring act. Similar to the `refer` action schema, the effect is formulated

```
HEADER:        s-attrib(Entity,λX·called(X,Name))
WHERE:         speaker(Speaker)
               hearer(Hearer)
               ref(Entity,Object)
CONSTRAINTS:   called(Object,Name)
EFFECTS:       bel(Hearer,goal(Speaker,mb(Speaker,Hearer,
                                  called(Object,Name))))
```

Figure 4.10: `s-attrib` schema (for `called`)

so that the hearer can recognize the intention to establish mutual belief even if it cannot established.

The third type of surface speech action, `s-attrib-rel`, is used to describe a relation between the referent and another object. An example for the `has` property (that describes an object that is 'part' of the referent) is shown in figure 4.11. The constraint and effect

```
HEADER:        s-attrib-rel(Entity,TEntity,λX·λY·has(X,Y))
WHERE:         speaker(Speaker)
               hearer(Hearer)
               ref(Entity,Object)
               ref(TEntity,Thing)
CONSTRAINTS:   has(Object,Thing)
EFFECTS:       bel(Hearer,goal(Speaker,mb(Speaker,Hearer,
                                  has(Object,Thing))))
```

Figure 4.11: `s-attrib-rel` schema (for `has`)

for this action are similar to those for the `s-attrib` schemas.

In Heeman's system, the referring act has only one effect corresponding to the top-level `refer` action. He made this decision to simplify his implementation. Obviously, in our theory the referring act has one main effect and several side-effects, one for each attribute. Our implementation takes the multiple effects into account, and allows each one to be reasoned about individually. We describe this process in chapter 6.

4.5 An Example

This section presents an example that illustrates how a speaker constructs a referring expression plan from his beliefs, and how he communicates the plan via surface speech acts. Then, we illustrate how a hearer infers the plan from the surface speech acts that she observes, and how her mental state is changed.

4.5.1 Generating a Referring Expression

Assume that the system (acting as speaker) believes the following propositions:

```
category(inter1,intersection).
category(inter2,intersection).
called(inter2,'Lowell Street').
has(inter2,sign1).
category(sign1,sign).
```

Also assume that the system believes the salience hierarchies shown in section 4.2.2, and that the system has a confidence threshold of 2. The system would like the user, the hearer, to be able to identify `inter2`, within the context of `inter1`, `inter2`, and `sign1`, and so constructs a referring expression that uniquely identifies `inter2`. The constructor first chooses the head noun, `intersection`, from the object's category property and creates an initial candidate set that includes both `inter1` and `inter2`. The constructor now attempts to modify the head noun (because the candidate set does not contain a single object), and comes up with two possibilities. It could use either the `called` property or the `has` property. By using the former, the plan contains the fewest primitive actions, exceeds the confidence threshold, and uniquely specifies the referent. The constructed plan derivation is shown in figure 4.12, and results in the following surface speech acts:

```
s-attrib(entity(1,inter2),λX·called(X,'Lowell Street'))
s-attrib(entity(1,inter2),λX·category(X,intersection))
s-refer(entity(1,inter2))
```

which could be realized as

> *The Lowell Street intersection.*

4.5.2 Understanding a Referring Expression

Now we reverse our point of view and consider the system, acting as hearer, observing the above surface speech acts. The hearer has no beliefs about the referent (from assumption (4.1)), but she does have beliefs about the salience hierarchies. We will assume the hearer has the same beliefs about salience as the speaker does, but has a different confidence threshold, a value of 5. To infer the speaker's plan, the system first recognizes the referring plan that accounts for the observed surface speech acts. This plan is shown in figure 4.13 with the WHERE predicates instantiated, and all co-referring variables unified.

The second step is to evaluate the plan. Since the system has no knowledge of the referent, it creates a new object name (`object1`) and instantiates the `Object` variable with it. This instantiation has the effect of making all the properties of `Object` used in the plan be properties of the new object `object1`. Now the system evaluates the constraints p20: refer(entity(1,inter2),3)

    s-refer(entity(1,inter2))
    describe(entity(1,inter2),3)
        context-current([sign1,inter1,inter2])
        headnoun([sign1,inter1,inter2],entity(1,inter2),[inter1,inter2],
                [category(inter2,intersection)],1)
            s-attrib(entity(1,inter2),λX $\cdot$ category(X,intersection))
                CONSTRAINT: *category(inter2,intersection)*
                EFFECT: *bel(user,goal(system,mb(system,user,category(inter2,intersection))))*
            subset([sign1,inter1,inter2],λX $\cdot$ ab(system,user,category(X,intersection)),
                [inter1,inter2])
            ref(entity(1,inter2),inter2)
            [category(inter2,intersection)]=[category(inter2,intersection)]
            confidence-headnoun(1,entity(1,inter2),intersection)
        modifiers(entity(1,inter2),[inter1,inter2],[category(inter2,intersection)],1,3)
            modifier(entity(1,inter2),[inter1,inter2],[inter2],[category(inter2,intersection)],
                [called(inter2,'Lowell Street'),category(inter2,intersection)],2)
                salient-attrib(entity(1,inter2),λX $\cdot$ called(X,'Lowell Street'),
                    [category(inter2,intersection)],[called(inter2,'Lowell Street'),
                    category(inter2,intersection)])
                s-attrib(entity(1,inter2),λX $\cdot$ called(X,'Lowell Street'))
                    CONSTRAINT: *called(inter2,'Lowell Street')*
                    EFFECT: *bel(user,goal(system,mb(system,user,called(inter2,'Lowell Street'))))*
                subset([inter1,inter2],λX $\cdot$ ab(system,user,called(X,'Lowell Street')),[inter2])
                confidence-attrib(2,entity(1,inter2),λX $\cdot$ called(X,'Lowell Street'))
            confidence-add(3,1,2)
            p53: modifiers(entity(1,inter2),[inter2],[called(inter2,'Lowell Street'),
                category(inter2,intersection)],3,3)
                CONSTRAINT: *p55: [inter2] = [inter2]*
                confidence-exceed(entity(1,inter2),3)
                null

EFFECT: *bel(user,goal(system,identify(user,system,entity(1,inter2))))*

Figure 4.12: Plan derivation for *The Lowell Street intersection.*

```
p20: refer(entity(1,Object),OvrConf)
  s-refer(entity(1,Object))
  describe(entity(1,Object),OvrConf)
    context-current(Context)
    headnoun(Context,entity(1,Object),Cand,UsedPred,Conf)
      s-attrib(entity(1,Object),λX·category(X,intersection))
        CONSTRAINT:  category(Object,intersection)
        EFFECT:  bel(system,goal(user,mb(user,system,category(Object,intersection
      subset(Context,λX·ab(user,system,category(X,intersection)),Cand)
      ref(entity(1,Object),Object)
      UsedPred=[category(Object,intersection)]
      confidence-headnoun(Conf,entity(1,Object),intersection)
    modifiers(entity(1,Object),Cand,UsedPred,Conf,OvrConf)
      modifier(entity(1,Object),Cand,NewCand,UsedPred,NewUsedPred,ModConf)
        salient-attrib(entity(1,Object),λX·called(X,'Lowell Street'),
            UsedPred,NewUsedPred)
        s-attrib(entity(1,Object),λX·called(X,'Lowell Street'))
          CONSTRAINT:  called(Object,'Lowell Street')
          EFFECT:  bel(system,goal(user,mb(user,system,called(Object,'Lowell Street'))))
        subset(Cand,λX·ab(user,system,called(X,'Lowell Street')),NewCand)
        confidence-attrib(ModConf,entity(1,Object),λX·called(X,'Lowell Street'))
      confidence-add(OvrConf,Conf,ModConf)
      p53: modifiers(entity(1,Object),NewCand,NewUsedPred,OvrConf,OvrConf)
        CONSTRAINT:  NewCand = [Object]
                     p56: confidence-exceed(entity(1,Object),OvrConf)
        null

  EFFECT:  bel(system,goal(user,identify(system,user,entity(1,Object))))
```

Figure 4.13: Recognized referring expression plan

and mental actions. This results in `Cand` and `NewCand` being instantiated to `[object1]` by the `subset` actions. `UsedPred` is instantiated to `[category(object1,intersection)]`, and `NewUsedPred` to `[called(object1,'Lowell Street'), category(object1,intersection)]` indicating that both the `called` and `category` properties have been used in the description. Furthermore, the constraints from the two `s-attrib` actions cause the system to add the two beliefs:

```
bel(user,category(object1,intersection)).
bel(user,called(object1,'Lowell Street')).
```

to its mental state. The mental actions that compute confidence are also evaluated. From the salience hierarchies shown in section 4.2.2, `Conf` gets instantiated to 1, `ModConf` to 2, and the overall confidence, `OvrConf`, to 3. Notice that this value does not exceed the threshold value of 5, and therefore, evaluation fails at the `confidence-exceed` constraint of the `null` terminated `modifiers` schema.

```
p20: refer(entity(1,object1),3)
  s-refer(entity(1,object1))
  describe(entity(1,object1),3)
    context-current([object1])
    headnoun([object1],entity(1,object1),[object1],[category(object1,intersection)],1)
      s-attrib(entity(1,object1),λX·category(X,intersection))
        CONSTRAINT: category(object1,intersection)
        EFFECT: bel(system,goal(user,mb(user,system,category(object1,intersection))))
      subset([object1],λX·ab(system,user,category(X,intersection)),[object1])
      ref(entity(1,object1),object1)
      [category(object1,intersection)]=[category(object1,intersection)]
      confidence-headnoun(1,entity(1,object1),intersection)
    modifiers(entity(1,object1),[object1],[category(object1,intersection)],1,3)
      modifier(entity(1,object1),[object1],[object1],[category(object1,intersection)],
            [called(object1,'Lowell Street'),category(object1,intersection)],2)
        salient-attrib(entity(1,object1),λX·called(X,'Lowell Street'),
              [category(object1,intersection)],[called(object1,'Lowell Street'),
              category(object1,intersection)])
        s-attrib(entity(1,object1),λX·called(X,'Lowell Street'))
          CONSTRAINT: called(object1,'Lowell Street')
          EFFECT: bel(system,goal(user,mb(user,system,called(object1,'Lowell Street'))))
        subset([object1],λX·ab(system,user,called(X,'Lowell Street')),[object1])
        confidence-attrib(2,entity(1,object1),λX·called(X,'Lowell Street'))
      confidence-add(3,1,2)
      p53: modifiers(entity(1,object1),[object1],[called(object1,'Lowell Street'),
            category(object1,intersection)],3,3)
        CONSTRAINT: [object1] = [object1]
                    p56: confidence-exceed(entity(1,object1),3)
        null

  EFFECT: bel(system,goal(user,identify(system,user,entity(1,object1))))
```

Figure 4.14: Inferred plan derivation for *The Lowell Street intersection.*

The evaluated plan is shown in figure 4.14. The system goes on to reason about the effects of the plan. Since the system believes the speaker is rational, it can infer its own beliefs about the properties of `object1`. Specifically, the system now believes:

```
bel(user,category(object1,intersection)).
bel(user,called(object1,'Lowell Street')).
```

The final effect of the plan is for the system to know how to identify the referent. But this effect cannot occur because of the error in confidence evaluation. The hearer is not confident that she will be able to identify the referent. In the next chapter we discuss the options the hearer has to get this error corrected.

4.6 Summary

In this chapter we presented a model of how a referring expression can be generated and understood, within the planning paradigm, for which the referent is not mutually known. Although it was not explicitly stated, we view referring in this context as a speech act akin to informing. The model accounts for how the propositional content of a referring plan is generated and understood within the theory of speech acts.

The model accounts for the two issues we set out to resolve. First, how a suitable referring expression can be built with the underlying intention of having the hearer know how to identify the referent, eventually. And second, how the notion of confidence in the adequacy of such an expression can be incorporated into the expression's plan derivation.

One application of this model is in the direction-giving domain where the direction giver and recipient have no mutual knowledge of the paths and places along a route. The direction giver, during a direction-giving dialogue, attempts to make the most salient, and therefore helpful, references to paths and places that he can. But because the recipient has never heard of these paths and places, and because she cannot actually execute the directions while hearing them, she must judge how adequate the directions are. The adequacy of referring expression subplans (of a route description plan) is of primary importance to determining the adequacy of route plans. So, if the hearer is not confident about the references made, then she cannot be confident about the directions.[12]

What does the recipient do if she is not confident about the adequacy of the directions? She attempts to negotiate or cooperate with the direction giver about the directions until she is confident. In the next two chapters we explore our model of collaboration on direction-giving.

[12]This is the only type of error we address in route description plans (see section 3.5.3).

Chapter 5

Elaboration

5.1 Introduction

This chapter presents our model of how two agents in a conversation cooperate to elaborate a referring expression plan. The model is similar to Heeman's model for making clarifications of referring expressions, but since we are dealing with a different type of referring action, we need a different process for repairing inadequate referring expressions. Heeman's clarifications are intended to constrain a referent's description more (or less) than it is already, whereas an *elaboration* is intended to provide additional description to an already uniquely defined referent.[1]

An agent has two courses of action when she believes a referring expression is inadequate. She can either *suggest* a way for another agent to elaborate it, or she can actually *elaborate* it herself. Section 5.2 explains the conversational moves of suggestion and elaboration, and section 5.3 casts these moves into the planning paradigm.

The discourse action schemas that we define below are only half of the solution. We must also account for the intentional structure of the discourse, which means modelling how the judgement of a plan (or action) influences the adoption of discourse goals that further the conversation. This second half of the model is presented in the next chapter, while in this chapter, we deal with elaboration in isolation.

5.2 Suggesting and Elaborating

In Clark and Wilkes-Gibbs's (1986) model of collaboration on referring expressions, participants use conversational moves that express a judgement on the success of a referring

[1]By the term *clarification*, Heeman intended to encompass all forms of judgement and refashioning (including elaboration) to referring expressions. We prefer to view clarification as a way to make an expression more clear by further disambiguating the referent, and elaboration as a way of adding more information, not necessarily more detail, to a description.

expression (acceptance, postponement, or rejection), and moves that refashion an expression (expansion or replacement). Heeman and Hirst (1992), in their computational implementation, use the term *clarification* to refer to these moves because they serve to make the expression more clear. In their model of referring, an expression is judged to be unsuccessful, requiring clarification, if it is overconstrained or underconstrained.

Although Clark and Wilkes-Gibbs considered referring to mutually known objects, the conversational moves that they observed are equally applicable to referring in our context. In our model of referring, expressions are never overconstrained or underconstrained from the point of view of the hearer. So, in this sense, they are always clear. However, as we discussed in chapter 4, an agent can be concerned about the inadequacy of a referring expression.

When the responder to a referring action is not confident in the adequacy of the plan, she first *postpones* her judgement of its ultimate acceptability because, although she finds it acceptable so far, she requires more information. By using this action, the responder communicates to the initiator that she is not confident in the adequacy of the current expression. The next step in Clark and Wilkes-Gibbs's model is for one of the participants to refashion the expression. Since the responder has no knowledge of the referent, other than that contained within the referring expression itself, she cannot refashion the expression herself. The initiator of the expression does have the ability to refashion the expression and can do so by *expanding* it.[2]

Although the responder cannot expand an expression herself, she does have the ability to help the initiator by suggesting a good way to expand it. We propose that *suggestion* is a conversational move in which an agent suggests a new modifier (of the referent) that she deems would increase her confidence in the expression's adequacy if the expression was expanded to include that modifier. For example, if the responder was not confident about the adequacy of *the schnauzer*, she might suggest that the initiator use the size of the canine (as well as its breed), by asking *What size is it?*. From this suggestion the initiator could expand the expression to *the miniature schnauzer*. Note that, in our sense, a suggestion is an illocutionary act of questioning. Along with actually suggesting a way to expand a plan, the suggester is asking whether or not the referent has the suggested property.

We restrict the size of a suggestion to *one* modifier for several reasons, even though the addition of this one modifier may not make the plan adequate in the eyes of the responder. First, it seems appropriate to not inundate another agent with a large number of suggested modifiers, and to, instead, proceed one step at a time. And second, since we are modelling collaborative dialogue, the hope behind making a suggestion is that the other agent will

[2]In Clark and Wilkes-Gibbs's model, an expansion always follows a postponement, and a replacement always follows a rejection. In our model we are only concerned with adding new modifiers to an expression, so we consider only the postponement and expansion pair of moves.

expand the plan with extra (non-suggested) information.

Since an agent suggests only one modifier of the referent, she would like to suggest the most salient property that is not already used in the expression to potentially increase her confidence the most. Therefore, she accesses her property salience hierarchy to find what she believes to be the most salient property for the type of object that has not been used in the plan already.

Depending on whether the responder makes a suggestion or not, the initiator has two options when expanding a plan. If a suggestion was not made, then he can expand the plan according to his own beliefs about the referent's properties and their salience. On the other hand, if a suggestion was made, he can attempt to expand the plan by using the suggested modifier. The first option is simple to perform because the plan constructor can just continue from where it left off before. The second option is more involved and we will consider it further.

The first step to expand a plan according to a suggestion is to construct the expansion. Using a suggestion is more involved for the plan constructor because it must build a plan with a specific set of actions (those corresponding to the suggested modifier) in its yield. Up until now, we have described the plan constructor as constructing a plan and returning a set of primitive actions, the plan's yield. We would like to force the constructor to build a plan that has the suggested actions in its yield; therefore, we modified the plan constructor in the following way. The plan constructor is given a set of suggested actions as a parameter. A new planning heuristic, the *suggestion heuristic*, which takes precedence over Heeman's *best-first heuristic*,[3] considers an action whose yield is contained in the suggested set of actions first. Thus, the constructor attempts to add actions that correspond to the suggested modifier first. If it cannot add these actions, then it considers other actions in the order specified by the *best-first heuristic*. The result is a plan that has the suggested actions in its yield, if possible.

The response to a suggestion depends, obviously, on whether or not the suggestion was used to expand the plan. There are three possible cases, shown below, each of which generates a different response.

1. The suggestion was used, and was enough to make the initiator confident about the adequacy of the referring plan. In this case, the response simply affirms that the plan was expanded with the suggestion.

2. The suggestion was used, but it was not enough to make the initiator confident, who, therefore, expanded the plan with additional modifiers. Here, the initiator responds by affirming that the suggestion was used, and by informing that additional modifiers were also used.

[3] Recall that the *best-first heuristic* considers the action with the smallest yield (of primitive actions) first.

3. It was not possible to use the suggestion (or the suggestion was inappropriate), so the initiator expanded the plan using other modifiers. The response, in this case, rejects or denies the suggestion, and also informs the suggester of the expansion that was actually made.

5.3 Discourse Action Schemas

Like Heeman, we model conversational moves as meta-actions (meta-plans) that act on referring expression plans. The responder first expresses her judgement (based on her confidence) of the referring expression plan by using the `accept-plan` or `postpone-plan` discourse actions. If she postpones her ultimate judgement, then she can either suggest, using `suggest-expand-plan`, a way to elaborate the plan, or wait for her partner to elaborate the plan with an `expand-plan` action.[4]

We do not intend these discourse actions to account for all the ways a referring expression may be elaborated, or all the ways a suggestion may be formed, but we believe this approach is feasible and versatile enough to be extended. The schemas should be seen as a logical extension of those defined by Heeman, with the ultimate goal of developing a model that handles many types of referring acts that occur in collaborative dialogue.

For example, the `expand-plan` schemas defined below are used only by the initiator of a referring expression because of assumption (4.1). Since the responder has no knowledge of the initiator's intended referent, other than that obtained from the expression itself, she cannot expand the expression. If assumption (4.1) were to be relaxed, thus allowing the responder to have some knowledge about the referent, she could use these schemas too, because they are general enough to be used by either participant in a conversation.

Likewise, the initiator could use the `suggest-expand-plan` action to see if the responder would like the referring plan to be expanded in his suggested way.

5.3.1 accept-plan

The discourse action `accept-plan`, shown in figure 5.1, is used by the speaker to establish the mutual belief that a plan achieves its goal. The constraint makes sure the speaker believes that the subplan[5] `SubPlan` of the plan achieves its goal, and the decomposition is the surface speech act `s-accept`. The effect of the action is that the hearer will believe that the speaker has the goal that it be mutually believed that the subplan of the plan

[4] If she has knowledge of the referent, she could also expand the plan herself, and inform the initiator of the expansion. The option is available to the recipient but it is never taken, because the recipient has no knowledge of the referent.

[5] The subplan and its goal that the `accept-plan` move operates on is the subplan in the agent's focus of attention, which is found by the WHERE-clause `focus-current`. All of the discourse actions in this chapter operate on the current subplan in focus, the tracking of which is described in section 6.2 of the next chapter.

```
HEADER:          accept-plan(Plan)
WHERE:           speaker(Speaker)
                 hearer(Hearer)
                 focus-current(focus(Plan,SubPlan,SubGoal))
CONSTRAINTS:     achieve(Plan,SubPlan,SubGoal)
DECOMPOSITION:   s-accept(Plan)
EFFECTS:         bel(Hearer,goal(Speaker,mb(Speaker,Hearer,
                       achieve(Plan,SubPlan,SubGoal))))
```

Figure 5.1: `accept-plan` schema

achieves its goal. The effect is formulated in this way, as are all the effects of the discourse action schemas, because the speaker cannot directly affect the beliefs of the hearer, and must, therefore, express his intentions in a recognizable manner.

5.3.2 postpone-plan

The discourse action `postpone-plan`, shown in figure 5.2, is used by the speaker when he is not confident in the adequacy of a referring expression. By using this action, the

```
HEADER:          postpone-plan(Plan)
WHERE:           speaker(Speaker)
                 hearer(Hearer)
                 focus-current(focus(Plan,SubPlan,SubGoal))
CONSTRAINTS:     error(Plan,SubPlan,ErrorNode)
                 constraint(Plan,ParentNode,ErrorNode)
                 yield(Plan,ParentNode,[ ])
                 content(Plan,ParentNode,Content)
                 Content = modifiers(Entity,Cand,UsedPred,Conf,OvrConf)
DECOMPOSITION:   s-postpone(Plan)
EFFECTS:         bel(Hearer,goal(Speaker,mb(Speaker,Hearer,
                       error(Plan,SubPlan,ErrorNode))))
```

Figure 5.2: `postpone-plan` schema

speaker informs the hearer that the plan is inadequate and that elaboration is required. The constraints first determine the node, `ErrorNode`, of the refer `SubPlan` within `Plan`, where the 'error' occurs. Second, they ensure that `ErrorNode` is a constraint of a node `ParentNode`. Third, they make sure that `ParentNode` has no primitive actions (i.e., that it has a `null` decomposition). Fourth and fifth, they find the content of `ParentNode` and make sure that it is an instance of a `modifiers` schema. In other words, the constraints require that the error node be the constraint of a `modifiers` action where expansion is possible.

The decomposition is the single surface speech action `s-postpone`. The effect of the `postpone-plan` action is that the hearer will believe that the speaker has the goal of establishing the mutual belief that the referring plan has an error at `ErrorNode`. This action has the additional side-effect, or perlocutionary effect, on the initiator of the referring expression of raising his confidence threshold. So, the initiator will no longer believe that he is confident in the adequacy of the current plan.

5.3.3 suggest-expand-plan

The discourse action `suggest-expand-plan`, shown in figure 5.3, is used by the speaker to suggest a way to expand the referring expression plan. The constraints are identi-

HEADER:	`suggest-expand-plan(Plan)`
WHERE:	`speaker(Speaker)`
	`hearer(Hearer)`
	`focus-current(focus(Plan,SubPlan,SubGoal))`
CONSTRAINTS:	`error(Plan,SubPlan,ErrorNode)`
	`constraint(Plan,ParentNode,ErrorNode)`
	`yield(Plan,ParentNode,[ ])`
	`content(Plan,ParentNode,Content)`
	`Content = modifiers(Entity,Cand,UsedPred,Conf,OvrConf)`
DECOMPOSITION:	`salience-suggest-actions(Plan,SubPlan,Entity,Actions)`
	`s-suggest(Plan,Actions)`
EFFECTS:	`bel(Hearer,goal(Speaker,mb(Speaker,Hearer,`
	`suggest(Plan,SubPlan,Actions))))`

Figure 5.3: `suggest-expand-plan` schema

cal to those of `postpone-plan`. The first step of the decomposition, the mental action `salience-suggest-actions`, accesses the speaker's property salience hierarchy and determines the property (that is not already used in `SubPlan` of `Plan`) that is the most salient for the category of the referent. It returns a set of 'skeleton' surface speech actions, `Actions`, that are used to inform the hearer of the suggestion by using the surface speech action `s-suggest` in the second step. The effect of this action is for the hearer to believe that the speaker has the goal of establishing the mutual belief that these actions constitute a suggestion.

5.3.4 expand-plan

The discourse action `expand-plan` is used by the speaker to replace a plan with a new one by elaborating on a referring expression within it. Because the modifiers in a referring expression plan are constructed recursively by using `modifiers` schemas, new modifiers must be added at the final `modifiers` action that terminates the recursion.

We define two `expand-plan` schemas. The first, shown in figure 5.4, is used to expand a referring expression plan by using the most salient property that has not already been

HEADER:	`expand-plan(Plan)`
WHERE:	`speaker(Speaker)`
	`hearer(Hearer)`
	`focus-current(focus(Plan,SubPlan,SubGoal))`
CONSTRAINTS:	`error(Plan,SubPlan,ErrorNode)`
	`constraint(Plan,ParentNode,ErrorNode)`
	`yield(Plan,ParentNode,[ ])`
	`content(Plan,ParentNode,Content)`
	`Content = modifiers(Entity,Cand,UsedPred,Conf,OvrConf)`
	`not(confidence-exceed(Entity,Conf))`
	`not(suggest(Plan,SubPlan,SActs))`
DECOMPOSITION:	`construct(modifiers(Entity,Cand,UsedPred,Conf,NewConf),` `Expansion,[ ],Actions)`
	`substitute(Plan,ParentNode,Expansion,NewPlan,Actions)`
	`evaluate(NewPlan)`
	`s-actions(Plan,Actions)`
EFFECTS:	`bel(Hearer,goal(Speaker,mb(Speaker,Hearer,` `replace(Plan,NewPlan))))`

Figure 5.4: `expand-plan` schema

used in the plan. This action is only used if a suggestion has not been made. The first five constraints are the same as those of `postpone-plan`: they determine the 'error' node and make sure it is a constraint of a `modifiers` action with a `null` decomposition. The fifth constraint also instantiates the final candidate set, used-predicate set, and confidence sum that will be used to construct the new modifiers. The sixth constraint makes sure that the confidence threshold for the refer plan has not been exceeded, meaning that the plan is deemed inadequate and requires expansion. The final constraint makes sure that no suggestion to `Subplan` of `Plan` has been made.

The decomposition specifies how a new plan can be built. First, the plan constructor is called to construct an instance of a `modifiers` subplan starting where the constructor ended before (by supplying to it the final candidate set, used-predicate set, and confidence sum). Obviously, the constructor will select the most salient properties possible until the confidence threshold is once again exceeded. Second, the constructed subplan, `Expansion`, is substituted for the old `modifiers` subplan, creating the new plan `NewPlan`. Third, the plan is evaluated to make sure it is still a valid plan. This step is necessary because the plan constructor (called in the first step) is ignorant of how the expansion will interact with the rest of the plan. Additionally, the evaluation re-unifies the overall confidence value in the whole plan derivation. Finally, the surface speech action `s-actions` is used to inform of hearer of the new surface speech actions that have been added to the plan.

The effect of the plan is that the hearer will believe that the speaker has the goal that it be mutually believed that the old referring expression plan be replaced by the newly elaborated plan.

The second `expand-plan` action, shown in figure 5.5, is used by an agent to expand a referring expression plan after another agent has made a suggestion. The suggested surface speech actions are used to construct a new `modifiers` action to append to the referring expression plan.

```
HEADER:          expand-plan(Plan)
WHERE:           speaker(Speaker)
                 hearer(Hearer)
                 focus-current(focus(Plan,SubPlan,SubGoal))
CONSTRAINTS:     error(Plan,SubPlan,ErrorNode)
                 constraint(Plan,ParentNode,ErrorNode)
                 yield(Plan,ParentNode,[ ])
                 content(Plan,ParentNode,Content)
                 Content = modifiers(Entity,Cand,UsedPred,Conf,OvrConf)
                 not(confidence-exceed(Entity,Conf))
                 suggest(Plan,SubPlan,SActs)
DECOMPOSITION:   construct(modifiers(Entity,Cand,UsedPred,Conf,NewConf),
                           Expansion,SActs,Actions)
                 substitute(Plan,ParentNode,Expansion,NewPlan,Actions)
                 evaluate(NewPlan)
                 respond-to-suggestion(Plan,SActs,Actions)
EFFECTS:         bel(Hearer,goal(Speaker,mb(Speaker,Hearer,
                       replace(Plan,NewPlan))))
```

Figure 5.5: `expand-plan` schema

The first six constraints of this schema are identical to those in the first schema. The seventh constraint makes sure that a suggestion to the subplan `Subplan` of the plan has been made. The `suggest(Plan,SubPlan,SActs)` belief is realized as a perlocutionary effect of inferring and accepting a `suggest-expand-plan` action. This action has the same effect as the first `expand-plan` action.

The first step of the decomposition constructs a `modifiers` action by attempting to use the suggested actions. The next two steps are similar to those of the first expansion move. The final step is an action that responds to the suggestion via surface speech acts. The actual response depends on the set of suggested actions and the actual yield of the expansion.

In order to respond to suggestions, we define three `respond-to-suggestion` action schemas. Only one of these schemas is applicable at a time, depending on how the expansion was constructed. In the first schema, shown in figure 5.6, the constraint makes sure that the set of suggested actions equals the set of expansion actions meaning that the suggestion, and only the suggestion, was used in the expansion. In this case the decomposition is the single surface speech action `s-affirm` that affirms that the suggestion was taken.

```
HEADER:         respond-to-suggestion(Plan,SActs,Acts)
CONSTRAINTS:    seteq(SActs,Acts)
DECOMPOSITION:  s-affirm(Plan,SActs)
```

Figure 5.6: `respond-to-suggestion` schema

The second schema, figure 5.7, is used when the plan is expanded with the suggested actions, and other additional actions. The constraint makes sure that the suggested actions are a proper subset of the expansion actions.

```
HEADER:         respond-to-suggestion(Plan,SActs,Acts)
CONSTRAINTS:    psubset(SActs,Acts)
DECOMPOSITION:  partition(Acts,SActs,OtherActs)
                s-affirm(Plan,SActs)
                s-actions(Plan,OtherActs)
```

Figure 5.7: `respond-to-suggestion` schema

The decomposition has a mental action which partitions the expansion actions into the suggested actions and the other actions, `OtherActs`. Then, with two surface speech acts, it affirms that the suggestion was used, and that the other actions were also added to the plan.

The third and final response to a suggestion occurs when the suggestion cannot be used. The constraint of the schema, figure 5.8, makes sure that the set of expansion actions does not contain the suggested actions. The decomposition, containing two surface speech acts,

```
HEADER:         respond-to-suggestion(Plan,SActs,Acts)
CONSTRAINTS:    not(subset(SActs,Acts))
DECOMPOSITION:  s-deny(Plan,SActs)
                s-actions(Plan,Acts)
```

Figure 5.8: `respond-to-suggestion` schema

denies that the suggested actions were used, and also informs how the plan was expanded.

5.4 Surface Speech Actions

We have taken Heeman's approach that there are surface speech actions for judgement and refashioning plans. They take as parameters the valid plan that is being accepted or elaborated, and a set of surface speech acts, if necessary. These actions are primitive, so they have no decomposition. They also have no constraints and no effects because these have been captured by the discourse actions defined above. Of the surface speech actions defined below, we use Heeman's definition for `s-accept`, `s-postpone`, and `s-actions`, and our own definition for `s-suggest`, `s-affirm`, and `s-deny`, since they are new.

The surface speech act `s-accept`, shown in figure 5.9, is the only step in the decomposition of the `accept-plan` action. Therefore, we could have formulated `accept-plan` as a surface speech act. The only parameter, `Plan`, is the plan that is being accepted. This act could be linguistically realized by an explicit acknowledgement such as *okay* or *yes*.

HEADER:	`s-accept(Plan)`

Figure 5.9: `s-accept` schema

The `s-postpone` surface speech act, shown in figure 5.10, is similar in form to `s-accept`. It could be linguistically realized by a tentatively voiced *okay?* Or, it could be realized together with another speech act such as `s-actions` or `s-suggest` defined below.

HEADER:	`s-postpone(Plan)`

Figure 5.10: `s-postpone` schema

The `s-actions` surface speech act, shown in figure 5.11, is used to signal that new primitive actions are being added to the plan. Both `expand-plan` actions determine the expansion, `Actions`, and might use `s-actions` as a step in their plan derivations. For example, the following surface speech act:

`s-actions(Plan,[s-attrib(Entity,`λ`X·colour(X,grey))])`

could be realized as *the grey one*.

HEADER:	`s-actions(Plan,Actions)`

Figure 5.11: `s-actions` schema

The `s-suggest` act, shown in figure 5.12, is used to express a suggested set of primitive actions, `Actions`, that correspond to the yield of a single `modifier` plan derivation. The parameter `Actions` can be a skeleton or template of possible actions (only some of the parameters would be uninstantiated), or can be a fully instantiated set of surface speech acts, like the actions provided to `s-actions`. For example, the following surface speech act:

`s-suggest(Plan,[s-attrib(Entity,`λ`X·size(X,Value))])`

with the parameter `Value` uninstantiated could be linguistically realized as the question *What size is it?* If `Value` were instead instantiated to, for instance, `big`, then the speaker could ask *Is it a big dog?*

There are two possible answers to an `s-suggest` surface act, the `s-affirm` act, and the `s-deny` act. The `s-affirm` act, shown in figure 5.13, is used to express that the suggested

HEADER:	`s-suggest(Plan,Actions)`

Figure 5.12: `s-suggest` schema

HEADER:	`s-affirm(Plan,Actions)`

Figure 5.13: `s-affirm` schema

actions `Actions` have been added to the referring plan. For example, the following surface speech act:

`s-affirm(Plan,[s-attrib(Entity,`λ`X·size(X,big))])`

could be realized as *Yes, it is big.* If this action were followed the `s-actions` act shown above, then the speaker could answer *Yes, it is big, and grey.*

The `s-deny` action, in figure 5.14, expresses the negative answer to a suggestion. The

HEADER:	`s-deny(Plan,Actions)`

Figure 5.14: `s-deny` schema

following surface speech act:

`s-deny(Plan,[s-attrib(Entity,`λ`X·size(X,big))])`

could be realized as *No, the dog isn't big.* When accompanied with an `s-actions` action, like the from above, the answer could be *No, but it is grey.*

5.5 An Example

5.5.1 Making a Suggestion

Consider the scenario, following from the example in section 4.5.2, where the system, acting as hearer, has just finished inferring the user's plan (see figure 4.14). The system has evaluated its confidence in the adequacy of the referring expression

The Lowell Street intersection

to be the value 3, which does not exceed the its confidence threshold of 5. The system has the following belief about the error as a result of inferring the plan in :

`error(p1,p20,p56)`

meaning that subplan `p20` (the inferred referring plan) of plan `p1` (the overall route description plan) has an error at node `p56`.

The system switches roles and becomes the speaker. As speaker, the system first adopts the goal of informing the user about the error and conveys this information with `postpone-plan`.

Now, since the system, having no other knowledge about the `object1`, cannot elaborate the plan itself, it tries to make a suggestion instead. The system adopts the following goal:

```
bel(user,goal(system,mb(system,user,suggest(p1,p20,Actions)))).
```

To achieve this goal, the constructor builds the plan derivation shown in figure 5.15. The first step has found the most salient property of an intersection (`object1`'s category) that has not already been used in the plan: the fact that an intersection can have a sign. The corresponding surface speech acts are used to instantiate the surface speech action

suggest-expand-plan(p1)
CONSTRAINT: *error(p1,p20,p56)*
constraint(p1,p53,p56)
yield(p1,p53,[ ])
content(p1,p53,modifiers(entity(1,object1),[object1],
[called(object1,'Lowell Street'),category(object1,intersection)],3,3))
modifiers(entity(1,object1),[object1],[called(object1,'Lowell Street'),
category(object1,intersection)],3,3)=modifiers(entity(1,object1),
[object1],[called(object1,'Lowell Street'),category(object1,intersection)],3,3)

salience-suggest-actions(p1,p20,entity(1,object1),Actions)
s-suggest(p1,Actions)

EFFECT: *bel(system,goal(user,mb(system,user,suggest(p1,p20,Actions))))*

where:
Actions = [s-attrib-rel(entity(1,object1),entity(N,Obj2),$\lambda X \cdot \lambda Y \cdot$has(X,Y)),
s-refer(entity(N,Obj2)),
s-attrib(entity(N,Obj2),$\lambda X \cdot$category(X,sign))]

Figure 5.15: Plan derivation of a suggestion

`s-suggest`.

This plan results in the following surface speech act:

```
s-suggest(p1, [s-attrib-rel(entity(1,object1),entity(N,Obj2),
                            λX·λY·has(X,Y)),
               s-refer(entity(N,Obj2)),
               s-attrib(entity(N,Obj2),λX·category(X,sign))])
```

which could be linguistically realized, together with the `s-postpone` act, as

Is there a sign?

5.5.2 Elaborating a Referring Expression

Now consider the scenario where the system, who constructed the initial referring expression *The Lowell Street intersection* (see figure 4.12), now acting as hearer, has just heard the suggestion made above. After inferring the speaker's plans (`postpone-plan` and `suggest-expand-plan`) that underlie the observed surface speech acts, the system believes that:

```
error(p1,p20,p55)
suggest(p1,p20,[s-attrib-rel(entity(1,inter2),entity(N,Obj2),
                             λX·λY·has(X,Y)),
                s-refer(entity(N,Obj2)),
                s-attrib(entity(N,Obj2),λX·category(X,sign))])
```

Node `p55` of subplan `p20` is an arbitrarily chosen constraint of the `modifiers` action that has a `null` decomposition. The system cannot identify the specific constraint that the user found in error from the surface speech act, but an arbitrary choice is all right because it indicates that there is an error somewhere in the `modifiers` action. Because the user informed the system of its 'error', the system also retracts its belief that the plan achieves its goal. The system also raises its confidence threshold to the value 5, because its old confidence threshold was, obviously, not high enough. The new value reflects the amount that confidence could be raised by adding one or two more modifiers to the referring expression.

The system switches roles to become the speaker and adopts the goal:

```
bel(user,goal(system,mb(system,user,replace(p1,NewPlan))))
```

with the intention of replacing the old plan with a new elaborated one.

The system then derives the plan derivation shown in figure 5.16. Plan construction results in `Expansion`, a `modifiers` plan derivation (shown in figure 5.17), that uses the suggestion, because the suggestion was good enough to raise confidence over the threshold. The `substitute` action then takes this expansion and substitutes it for the old `modifiers` action that had a `null` decomposition (node `p53` in the plan of figure 4.12). The resulting expanded referring expression plan, `NewPlan`, is shown in figure 5.18. Next, the constructor evaluates the new plan to make sure it is still valid, and finally uses the action `respond-to-suggestion`. This action decomposes into the single surface speech act:

```
s-affirm(p1, [s-attrib-rel(entity(1,inter2),entity(N,Obj2),
                            λX·λY·has(X,Y)),
              s-refer(entity(N,Obj2)),
              s-attrib(entity(N,Obj2),λX·category(X,sign))])
```

because the set of suggested actions is equal to the yield of the expansion. This act could be linguistically realized as

Yes, it has a sign.

This example shows the most simple result of a suggestion, a lone affirmation without extra expansion. We will illustrate the more complex results in a large example given in chapter 7.

```
expand-plan(p1)
  CONSTRAINT:  error(p1,p20,p55)
               constraint(p1,p53,p55)
               yield(p1,p53,[ ])
               content(p1,p53,modifiers(entity(1,inter2),[inter2],
                   [called(inter2,'Lowell Street'),category(inter2,intersection)],3,3))
               modifiers(entity(1,inter2),[inter2],[called(inter2,'Lowell Street'),
                   category(inter2,intersection)],3,3)=modifiers(entity(1,inter2),
                   [inter2],[called(inter2,'Lowell Street'),category(inter2,intersection)],3,3)
               not(confidence-exceed(entity(1,inter2),3))
               suggest(p1,p20,SActs)
  | construct(modifiers(entity(1,inter2),[inter2],3),Expansion,SActs,Actions)
  | substitute(p1,p50,Expansion,NewPlan,Actions)
  | evaluate(NewPlan)
  | respond-to-suggestion(p1,SActs,Actions)
  |   CONSTRAINT:  seteq(SActs,Actions)
  |   | s-affirm(p1,SActs)
  EFFECT:  bel(system,goal(user,mb(system,user,replace(p1,NewPlan))))

where:
Actions = SActs = [s-attrib-rel(entity(1,inter2),entity(N,Obj2),λX·λY·has(X,Y)),
                   s-refer(entity(N,Obj2)),
                   s-attrib(entity(N,Obj2),λX·category(X,sign))]
```

Figure 5.16: Plan derivation of an expansion

5.6 Summary

In this chapter we have presented a model, in the planning paradigm, that accounts for how two agents can cooperate to elaborate a referring expression plan. We have defined two discourse actions (`postpone-plan` and `expand-plan`) similar to Heeman's actions, and a third new action (`suggest-expand-plan`). An agent uses the same planning processes in collaboratively negotiating a referent as he uses in describing a route, or in referring to an object.

We do not intend these actions to model all the possible ways of making suggestions and elaborations. We believe they provide a feasible extendible platform onto which more actions could placed. In particular, we have modelled suggestion as a way of *asking* the another agent to *expand* a plan in a certain way. Obviously, not all suggestions are about expanding plans.

We have also proposed a new heuristic, the *suggestion heuristic*, for the plan constructor. This heuristic permits the constructor to choose suggested actions before other actions. It takes precedence over Heeman's *best-first heuristic* if any actions have been suggested.

In the previous chapter we accounted for how an initial referring expression was constructed and inferred, and in this chapter we accounted for how a single elaboration of an expression could be accomplished. Since this may not be enough to satisfy the hearer's

```
modifiers(entity(1,inter2),[inter2],[called(inter2,'Lowell Street'),
        category(inter2,intersection)],3,6)
  modifier(entity(1,inter2),[inter2],[inter2],[called(inter2,'Lowell Street'),
          category(inter2,intersection)],[has(inter2,sign1),called(inter2,'Lowell Street'),
          category(inter2,intersection)],3)
    salient-attrib-rel(entity(1,inter2),entity(4,sign1),λX·λY·has(X,Y),
         [called(inter2,'Lowell Street'),category(inter2,intersection)],[has(inter2,sign1),
         called(inter2,'Lowell Street'),category(inter2,intersection)])
    s-attrib-rel(entity(1,inter2),entity(4,sign1),λX·λY·has(X,Y))
      CONSTRAINT:  has(inter2,sign1)
      EFFECT:  bel(user,goal(system,mb(system,user,has(inter2,sign1))))
    ref(entity(4,sign1),sign1)
    subset([inter2],λX·ab(system,user,has(X,sign1)),[inter2])
    refer(entity(4,sign1),1)
      s-refer(entity(4,sign1))
      describe(entity(4,sign1),1)
        context-current([sign1,inter1,inter2])
        headnoun([sign1,inter1,inter2],entity(4,sign1),[sign1],[category(sign1,sign)],1)
          s-attrib(entity(4,sign1),λX·category(X,sign))
            CONSTRAINT:  category(sign1,sign)
            EFFECT:  bel(user,goal(system,mb(system,user,category(sign1,sign))))
          subset([sign1,inter1,inter2],λX·ab(system,user,category(X,sign)),[sign1])
          ref(entity(4,sign1),sign1)
          [category(sign1,sign)]=[category(sign1,sign)]
          confidence-headnoun(1,entity(4,sign1),sign)
        modifiers(entity(4,sign1),[sign1],[category(sign1,sign)],1,1)
          CONSTRAINT:  [sign1] = [sign1]
                       confidence-exceed(entity(4,sign1),1)
          null
    confidence-attrib-rel(3,entity(1,inter2),entity(4,sign1),λX·λY·has(X,Y))
  confidence-add(6,3,3)
  modifiers(entity(1,inter2),[inter2],[has(inter2,sign1),called(inter2,'Lowell Street'),
          category(inter2,intersection)],6,6)
    CONSTRAINT:  [inter2] = [inter2]
                 confidence-exceed(entity(1,inter2),6)
    null
```

Figure 5.17: A `modifiers` expansion for the referring expression plan of figure 4.12

refer(entity(1,inter2),6)

s-refer(entity(1,inter2))

describe(entity(1,inter2),6)

context-current([sign1,inter1,inter2])

headnoun([sign1,inter1,inter2],entity(1,inter2),[inter1,inter2],
[category(inter2,intersection)],1)

s-attrib(entity(1,inter2),λX $\cdot$ category(X,intersection))

CONSTRAINT: *category(inter2,intersection)*

EFFECT: *bel(user,goal(system,mb(system,user,category(inter2,intersection))))*

subset([sign1,inter1,inter2],λX $\cdot$ ab(system,user,category(X,intersection)),
[inter1,inter2])

ref(entity(1,inter2),inter2)

[category(inter2,intersection)]=[category(inter2,intersection)]

confidence-headnoun(1,entity(1,inter2),intersection)

modifiers(entity(1,inter2),[inter1,inter2],[category(inter2,intersection)],1,6)

modifier(entity(1,inter2),[inter1,inter2],[inter2],[category(inter2,intersection)],
[called(inter2,'Lowell Street'),category(inter2,intersection)],2)

salient-attrib(entity(1,inter2),λX $\cdot$ called(X,'Lowell Street'),
[category(inter2,intersection)],[called(inter2,'Lowell Street'),
category(inter2,intersection)])

s-attrib(entity(1,inter2),λX $\cdot$ called(X,'Lowell Street'))

CONSTRAINT: *called(inter2,'Lowell Street')*

EFFECT: *bel(user,goal(system,mb(system,user,called(inter2,'Lowell Street'))))*

subset([inter1,inter2],λX $\cdot$ ab(system,user,called(X,'Lowell Street')),[inter2])

confidence-attrib(2,entity(1,inter2),λX $\cdot$ called(X,'Lowell Street'))

confidence-add(3,1,2)

modifiers plan derivation (shown in figure 5.17)

EFFECT: *bel(user,goal(system,identify(user,system,entity(1,inter2))))*

Figure 5.18: Referring expression plan after an expansion

confidence, we need to account for how further elaborations can be performed. Furthermore, we must account for how reference fits into the construction and inference of route description plans. We discuss these issues in the next chapter, where we complete our model of collaborative behaviour with an intentional structure that controls the adoption of goals and the acceptance of plans.

Chapter 6

Modelling Collaboration

6.1 Introduction

A model of collaboration in discourse within the planning paradigm should account for how an agent both constructs plans and infers plans. In order to collaboratively build up a domain plan, agents must converse until they reach a mutual acceptance of the plan, and this involves both generating language and understanding language. Many previous systems that incorporate speech acts and intentionality with utterances only model one side of a conversation: generating or understanding (Cohen and Perrault, 1979; Allen and Perrault, 1980; Appelt, 1985b; Litman and Allen, 1987), while other models consider the plans and goals at a level lower than the surface utterance plans (Carletta, 1990a; Carletta, 1990b; Grosz and Sidner, 1990; Lochbaum, Grosz and Sidner, 1990; Lambert and Carberry, 1991). A few models have considered both generation and understanding (McRoy, 1993; Heeman and Hirst, 1992). Heeman and Hirst's system models both plan construction and inference while accounting for how speech acts correspond to an utterance. However, their system, because it is for generating and understanding referring expressions (with only one underlying intention), cannot handle domain plans with several goals, side-effects, and underlying intentions. Nevertheless, their model is extendable to a model for more general collaborative discourse.

In the previous three chapters we discussed how route descriptions, referring expressions for places and paths on the route, and judgements and elaborations of referring expressions are generated and understood within our plan-based model. These processes are similar to what Grosz and Sidner (Grosz and Sidner, 1986) call the linguistic structure of a discourse. That is, they account for the actual utterances in a conversation, but not how they are connected. What we have left out of the model so far is a theory of how the collaborative intentions of the participants influence the discourse. In this chapter we will describe an intentional structure that includes the adoption of goals and the mutual acceptance of their related plans, both of which are sanctioned by a collaborative state that an agent is in.

In Heeman and Hirst's model, an agent, while collaborating on the development of a referring expression, is in a *collaborative state* that includes two components: the intention to achieve the goal to refer, and the current referring plan that the participants are considering in order to achieve the goal. The collaborative state sanctions the adoption of goals to express judgement of the current plan and goals to refashion it. It also sanctions the acceptance of these meta-plans and of the referring plan, which allows the mental state of the agent to be updated and/or revised.

We use this model as a basis for a more comprehensive model of collaborative discourse that is not limited to having one main discourse intention. As a discourse progresses over time, many intentions arise and these are translated into goals that are achieved by discourse plans. Our model accounts for how a domain plan can have a main goal, and many subgoals and side-effects, and how two agents can collaboratively reason about these goals using meta-plans.

Since a domain plan can be potentially very large, an agent needs a way to focus on specific subplans within the plan. We suggest that an agent maintains a stack of subplans within his collaborative state, in addition to the two components described above. The subplan on the top of the stack is currently in focus and is the one that he uses for both the generation and understanding of judgements and refashionings. In section 6.2 we discuss the collaborative state, and in particular, how the focus stack is maintained.

As in Heeman's system, the collaborative state sanctions the adoption of goals for meta-plans that operate on the domain plan. Specifically, an agent can express judgement (accepting or postponing) of a subplan, can elaborate a subplan, or can make suggestions to expand a subplan. We discuss the adoption of goals in section 6.3.

The collaborative state also sanctions the mutual acceptance of plans. The acceptance process is driven by an agent's need to establish the mutual acceptance of the domain plan and discourse plans. In short, for a domain plan to be mutually acceptable, all of its subplans must also be mutually acceptable to all of the agents in a conversation. In section 6.4 we discuss the acceptance process.

We complete the theory with a discussion of the reasoning process that an agent undergoes. The basic process is similar to Heeman's process, in that an agent uses reasoning rules to determine what to do next. We define two rule 'operators'. The first,

$$\mathtt{B} \overset{B}{\Longleftarrow} \mathtt{C}_1 \ \& \ \mathtt{C}_2 \ \& \ \cdots \ \& \ \mathtt{C}_n,$$

allows the agent's mental state to be updated (or revised) with belief B if all of the conditions are true. The second,

$$\mathtt{G} \overset{G}{\Longleftarrow} \mathtt{C}_1 \ \& \ \mathtt{C}_2 \ \& \ \cdots \ \& \ \mathtt{C}_n,$$

allows goal G to be adopted if all of the conditions are true.

The actual reasoning rules, to be defined in the next three sections, fall into three categories:

1. collaborative activity rules,
2. goal adoption rules, and
3. a mutual acceptance rule.

As a collaborative dialogue progresses, an agent goes through a reasoning process and applies different types of rules in different circumstances. The main factor in determining which type of rule an agent attempts to apply is his current role, be it speaker or hearer. For example, when a speaker doesn't yet have a goal, he attempts to use a goal adoption rule; or, when a hearer has just inferred a plan, she attempts to apply the mutual acceptance rule. The reasoning process governs how the dialogue progresses and is described in section 6.5.

We are making the assumption that the participants in a conversation have mutual knowledge of many of the processes involved. These include: the shifting of the focus of attention, reasoning using the rules, constructing and inferring plans, and the acceptance process. This mutual knowledge can be considered part of the social contract that a participant is accountable to upon entering a conversation (Boden and Zimmerman, 1991).

6.2 The Collaborative State

To model collaborative discourse, we need a way to account for how discourse goals arise, and how these goals are achieved. Clark and Wilkes-Gibbs (1986) propose that the participants in a conversation have a *mutual responsibility* for the success of a referring action. Clark and Schaefer (1989) go on to say that the success of a discourse depends on the coordinated actions of the participants—the participants must act collectively. But how do they do this without direct access to each other's beliefs?

Heeman proposes that an agent engaged in collaborative activity is in a collaborative mental state. We agree with Heeman, but we must extend his notion of the collaborative state to deal with general collaborative discourse. We propose that the collaborative state includes three components:

1. the intention to achieve the main goal of the collaborative activity,
2. the current domain plan that the agents are considering in order to achieve the main goal, and
3. a stack of plans that is used to track the agent's focus of attention during the discourse.

In our system, the main goal of the discourse is to develop a mutually acceptable route description. The domain plans (route description plans as defined in chapter 3) that we consider are not domain plans in the classical sense, because they are closely linked to the communicative actions that convey the plans as a series of utterances. A classical domain plan in task-oriented dialogue is separate from the discourse level plans (Grosz and Sidner, 1990; Lambert and Carberry, 1991; Heeman, 1993). We feel that the distinction between domain plan and discourse plan is blurred when the domain task involves the communication of information.

The current domain plan, along with the focus stack (described in detail below), serves to coordinate the collaborative activity of the participants, and so the agents keep it in their common ground. An agent may not believe that every action in a plan contributes to its goal, or even that every subgoal in the plan is achieved, unlike in the *shared plans* of Grosz and Sidner (1990), and Lambert and Carberry (1991), where every action must contribute and be mutually acceptable. Hence an agent has beliefs about the validity of the current plan and its subplans, and has the intention of making these beliefs mutually believed.

Because the collaborative state contains the current domain plan and the subplan of the domain plan that is in focus, it sanctions the adoption of goals to express judgement of the subplan in focus, to refashion the subplan in focus, or to suggest a way to expand the subplan in focus. The collaborative state also sanctions the mutual acceptance of the subplan in focus. In this sense, the collaborative state is fundamental to controlling the progression of a collaborative discourse because it keeps the collaborative task in the foreground and in the agents' common ground.

6.2.1 Entering into the Collaborative State

The rule for entering into collaborative activity can be applied simply if the agent has constructed or inferred a route description plan. So, if an agent believes that the speaker has a plan for achieving the goal of the hearer knowing the route, then the agent enters into the collaborative state. The rule is as follows:

```
cstate(Speaker,Hearer,Plan,Goal) <=B=
           hearer(Hearer) &
           speaker(Speaker) &
           plan(Speaker,Plan,Goal) &
           Goal = knowroute(Hearer,Speaker,Route)
```

Although the agents know when to stop collaborating (i.e., when the plan is mutually accepted), there is no rule for exiting the collaborative state, because we are modelling only the portion of dialogue that occurs while two agents are collaborating.

6.2.2 The Focus Stack

As described above, the focus stack is part of the mental state of an agent engaged in collaborative activity. The stack is a list of subplans of plans that an agent has beliefs about, for which the top element is the current focus of attention. The focus stack is very important for organizing the discourse, because it controls which subplan is being judged, refashioned, operated on, or accepted. The subplan in focus may be a subplan of the domain plan, or may be a discourse plan because discourse actions are subjected to the same acceptance process as the domain plan is. But how does focus shift, that is, when is a subplan popped or pushed?

After consulting the plan schemas defined in previous chapters, one can see that a plan derivation can have many effects. All the effects are formulated in such a way that the hearer of the plan can recognize the intentions of the speaker, modelled as subgoals that the domain plan is intended to achieve. Each subgoal is intended to be achieved by the subplan derivation rooted at the action with the corresponding effect. For example, a referring subplan of a route description plan is a subplan with the goal of having the hearer know how to identify the referent; and, an `s-attrib` primitive action is a subplan with the goal of establishing the mutual belief of an attribute of the referent.

An element on the stack is a triple consisting of the domain plan name, the subplan name, and the goal of the subplan. Associated with each focus element is a belief about the validity of the subplan. Either the subplan achieves its goal, or it is in error. The general rule for pushing subplans onto the focus stack is as follows:

> *Push Rule:*[1] Push subplans as they are encountered by an inorder traversal of the plan derivation (top to bottom, left to right). Separate plan derivations are considered in the order in which they are generated or understood.

This rule means that the last subplan encountered will be the first one to be operated on. The general rule for popping a subplan from the focus stack is as follows:

> *Pop Rule:* Pop a subplan if it is mutually acceptable. A subplan is mutually acceptable if it is mutually believed that it achieves its goal, or if it implicitly achieves its goal.

When the stack is empty, the collaborative discourse is over, because the domain plan has been mutually accepted.

The focus stack contains only subplans that have not been mutually accepted yet. Once a subplan is mutually accepted it will not be considered again, unless a further modification to the domain plan renders the subplan invalid.

Even though the focus stack specifies a rigid order in which contributions should be accepted, it is a plausible theory because the resulting discourse structure is similar to what Clark and Schaefer (1989) have observed in their studies. Recursive clarification

[1]Obviously, we must assume that the agents in the conversation have mutual knowledge about how to shift focus of attention. These two rules can be considered part of the social contract that a participant is accountable to.

subdialogues occur because all of the subplans on the stack below the subplan in focus cannot be considered until the subplan in focus is mutually accepted and popped from the stack. When humans converse they would like to update their common ground in a orderly way. The focus stack allows for this orderly behaviour via a simple mechanism that the participants in a collaborative dialogue can use so that they can present and accept contributions with a minimum of misunderstanding. However, because of the rigid structure imposed by the focus stack, the consideration of subplans cannot be ordered differently. We continue this discussion in section 6.4.4, after describing the acceptance process below.

6.3 Adopting Goals

As described above, a speaker adopts goals in order to collaborate in achieving the main goal of the domain plan. These goals are achieved by planning the discourse actions defined in the previous chapter. An agent may adopt a goal by applying a goal adoption rule permitted by the collaborative state which he is in. Any goal that he adopts will be a goal to operate on the subplan of the domain plan or a discourse plan that has his focus of attention. There is one rule for each type of goal an agent may plan for.

The first rule, shown below, is used by the speaker to adopt the goal to inform the hearer that a subplan is invalid. The conditions specify that the subplan in focus (of the current plan of the collaborative activity) has an error at node `Node`, and that this is not already mutually believed. A postponement action can achieve this goal.

```
goal(Speaker,mb(Speaker,Hearer,error(Plan,SubPlan,Node))) <=G=
            hearer(Hearer) &
            speaker(Speaker) &
            cstate(Speaker,Hearer,Plan,Goal) &
            focus-current(focus(Plan,SubPlan,SubGoal)) &
            error(Plan,SubPlan,Node) &
            not(mb(Speaker,Hearer,error(Plan,SubPlan,Node)))
```

The second rule is for adopting a goal to inform the hearer that a subplan is acceptable. The conditions for applying the rule specify that the speaker believes the subplan in focus achieves its goal, and that this belief is not mutually known. To achieve this goal, the speaker can plan an acceptance move.

```
goal(Speaker,mb(Speaker,Hearer,achieve(Plan,SubPlan,SubGoal))) <=G=
            hearer(Hearer) &
            speaker(Speaker) &
            cstate(Speaker,Hearer,Plan,Goal) &
            focus-current(focus(Plan,SubPlan,SubGoal)) &
            achieve(Plan,SubPlan,SubGoal) &
            not(mb(Speaker,Hearer,achieve(Plan,SubPlan,SubGoal)))
```

A speaker uses the third rule to adopt a goal to replace the current plan (that has an error in the subplan in focus) with a new refashioned plan. The conditions are similar to those of the first rule, except that the speaker must believe that it is mutually believed that there is an error. Hence, the speaker cannot refashion an invalid plan until after he expresses his judgement of the plan.

```
goal(Speaker,mb(Speaker,Hearer,replace(Plan,NewPlan))) ⟸G
            hearer(Hearer) &
            speaker(Speaker) &
            cstate(Speaker,Hearer,Plan,Goal) &
            focus-current(focus(Plan,SubPlan,SubGoal)) &
            mb(Speaker,Hearer,error(Plan,SubPlan,ErrorNode))
```

The fourth and final rule is used to adopt a goal to make a suggestion. The conditions are almost identical to those of the previous rule. They additionally specify that the suggestion has not already been made.

```
goal(Speaker,mb(Speaker,Hearer,suggest(Plan,SubPlan,Actions))) ⟸G
            hearer(Hearer) &
            speaker(Speaker) &
            cstate(Speaker,Hearer,Plan,Goal) &
            focus-current(focus(Plan,SubPlan,SubGoal)) &
            mb(Speaker,Hearer,error(Plan,SubPlan,ErrorNode)) &
            not(mb(Speaker,Hearer,suggest(Plan,SubPlan,Actions)))
```

There is actually a fifth goal adoption rule, but it is different from the above rules. An agent needs a way to adopt the initial goal of describing a route, a goal that is adopted before the collaborative state is entered. The following rule is used to adopt the initial goal. It has conditions to specify that the goal is not already being planned for in a collaborative activity, and that the route to be described is `Route`.

```
goal(Speaker,knowroute(Hearer,Speaker,Route)) ⟸G
            hearer(Hearer) &
            speaker(Speaker) &
            not(cstate(Speaker,Hearer,_,knowroute(_,_,_))) &
            what-route(Route)
```

6.4 The Acceptance Process

Clark and Schaefer (1989), in describing their model of how participants contribute to a discourse, suggest that a basic unit of conversation is the contribution, which must be mutually accepted by all participants before it can be added to their common ground. The driving force behind collaborative discourse is the intention to have a proposed contribution mutually accepted before proceeding. In their work, they associated contributions with utterances, but we feel that this association is too restrictive because it does not reflect the intentional structure of a discourse as described by Grosz and Sidner (1986). We prefer to associate contributions with the underlying intentions of an utterance. In this way, a single utterance can have several contributions within it, all of which must be submitted to the acceptance process. A contribution is a single subplan intended to achieve a goal, and it may have subcontributions, because it may have subplans.

6.4.1 Accepting One Contribution

But how does the acceptance process work? That is, how does a contribution come to be mutually accepted? In short, if an agent believes that it is mutually believed that a plan achieves its goal, then the plan is mutually acceptable. Starting at the beginning of the process, an agent, the presenter, presents a contribution. The contribution is a plan (or a subplan of a plan) that he believes achieves its goal. In a collaborative dialogue there is no sense in presenting a plan that one believes is invalid. Since he does not know if the contribution is acceptable to the other agent, the presenter waits for her to express her judgement of the plan. If she expresses acceptance, then the presenter infers that she believes the plan achieves its goal. Thus he can infer the belief that it is mutually believed that the plan achieves its goal. This inference is the culmination of the acceptance process because it allows the use of the mutual acceptance rule (defined below) that allows the effects of the plan to occur, thus updating the agents' common ground.

On the other hand, if the hearer of the contribution rejects the plan, then the presenter infers that the hearer believes the plan is invalid. This action opens up a clarification subdialogue which serves to make the current contribution mutually acceptable. The actions used in the subdialogue are contributions that themselves must be accepted; but, once the original contribution is accepted by both agents (i.e., the hearer finally expresses her acceptance, as before), the plan is mutually accepted.

Now, from the point of view of the hearer, after inferring the presenter's initial presentation from his utterance, she judges whether or not the plan is valid. If so, she expresses her acceptance of the plan, and then, presupposing the presenter's acceptance of this acceptance plan, she can infer the belief that it is mutually believed that the plan achieves its goal. And thus, she can apply the mutual acceptance rule that allows the plan's effects to occur.

If, however, she believes the plan is invalid, she initiates a clarification subdialogue by expressing her judgement that there is an error in the plan. As above, the clarification subdialogue terminates when she believes that it is mutually believed that the plan achieves its goal. And this belief allows the mutual acceptance of the plan.

6.4.2 The Grounds for Acceptance

The acceptance process is applied to every contribution, whether it is a route description, a referring expression, an expression of judgement, or a refashioning. Even moves that express acceptance are contributions that must themselves be accepted by another move. But what constitutes grounds for acceptance? In our model, a plan is acceptable to an agent if he believes it achieves its goal, and for each type of plan, an agent uses different criteria to determine if it achieves its goal. Figure 6.1 shows, for each different plan type, the criteria used in the determination. A necessary condition for any plan to achieve its goal is for it to be understood, which means that it must be valid as determined by the evaluation process.

Plan type	**Criterion**
Route description	Plan is valid (all referring subplans are valid, and route leads from origin to destination)
Referring	Plan is valid (referent uniquely specified, and confident plan is adequate)
Plans having a goal to establish the mutual belief of some property	Agent believes the other agent believes the property (rationality assumption)
Judgement (except acceptance), refashioning, and suggestion	Implicitly achieve their goal (if understood)
Acceptance	Agent believes the plan it accepts achieves its goal

Figure 6.1: Criteria for plans to achieve their goals

As discussed in chapter 3, the only type of error in a route description plan that we consider is inadequate referring expressions. This means that a route description plan achieves its goal if all the referring expression subplans within the plan achieve their goals. In addition, the route must have a sequence of actions that lead one from the origin to the destination.

A referring expression plan achieves its goal if it uniquely specifies the referent according to an agent's beliefs, and if the agent is confident that the plan is adequate.

Referring plans have subgoals to establish the mutual belief of the properties of the referent. By assuming that the other agent is rational (both sincere and competent), an agent can believe that a subgoal (of this type) is achieved if he believes that the other agent believes that the property is true. (See section 4.3.2.)

All of the meta-plans (judgements, refashionings, and suggestions), except for the acceptance move, achieve their goals if they are understood. To understand a judgement is to know which constraint in the domain plan the speaker found in error, without necessarily agreeing with the error. To understand a refashioning is to recognize the refashioned plan, without necessarily finding it acceptable. And to understand a suggestion is to know the suggested modifier, without necessarily believing the suggestion is a good one. If a plan is not understood, it must be ill-formed, which can result from mis-hearing an utterance, or from an incorrect utterance. We do not address the issue of ill-formed plans and their repair, except to be able to identify the condition.

The reason for implicitly accepting these plans is drawn from Clark and Wilkes-Gibbs's (and, subsequently, Heeman's) work on the acceptance process for referring expressions. These moves are always achieved (and so accepted), and this keeps the current referring plan in the agents' focus, leading to an iterative acceptance process for the referring plan. If these moves were not implicitly accepted, then they would have to be refashioned, leading to a recursive process, something that Clark and Wilkes-Gibbs did not observe in their studies. We discuss this point below, after defining the acceptance rule.

The acceptance meta-plan, however, does not implicitly achieve its goal (the belief in the mutual belief that the plan being accepted achieves its goal). For this plan, an agent must believe that the accepted plan achieves its goal. This makes sure that when the mutual acceptance rule is applied to the acceptance plan, the agent believes the accepted plan achieves its goal before adopting the mutual belief of this fact.

6.4.3 The Mutual Acceptance Rule

Because the agents are collaborating, a simple acceptance of a plan is not good enough—an agent must believe that it is mutually believed that a plan achieves its goal before allowing the effects of the actions to occur. Therefore, we define the following reasoning rule, the *mutual acceptance rule*.[2] Its conditions specify that an agent must believe that it is mutually believed that the subplan in focus achieves its goal, `SubGoal`.

$$\texttt{SubGoal} \overset{B}{\Longleftarrow}$$

```
          hearer(Hearer) &
          speaker(Speaker) &
          focus-current(focus(Plan,SubPlan,SubGoal)) &
          mb(Speaker,Hearer,achieve(Plan,SubPlan,SubGoal))
```

When this rule is applied, an agent uses `SubGoal` to update his mental state. In the general case, this involves simply adopting the belief that `SubGoal` is true. If `SubGoal` is a mutual belief then it becomes part of the agents' common ground.[3]

The adoption of some beliefs causes the revision of other beliefs, which are the perlocutionary effects of the speech act. To adopt the mutual belief that a plan has an error in it, the agent retracts his belief that the plan achieves its goal (if he had that belief).

[2]Unlike Heeman, we define only one generic acceptance rule. In his system, he defines one acceptance rule for each type of goal an agent can adopt.

[3]We model an agent's adoption of a mutual belief of a proposition as his adoption of the belief of the proposition and his adoption of the belief that the other agent believes it. This weaker notion of mutual belief simplifies the belief module, and it is implemented as an inference rule for inferring mutual belief.

To adopt the belief that a new plan replaces an old one, belief revision takes place. First, the old plan is removed from the agent's mental state. Second, the `cstate` is updated with the new plan, which includes updating the focus stack. And, third, the plan is evaluated, and new beliefs about the validities of its subplans are adopted. The newly added subplans are reasoned about and pushed to the focus stack because they are new contributions in the discourse. Any subplans that were already mutually accepted need not be considered again.[4]

In Heeman and Hirst's model, moves that express judgement and moves that refashion are always accepted if they are understood. This decision was made because the agents are involved in a collaborative activity and have the intention to keep the current domain plan in their common ground, while coordinating their activity. Heeman and Hirst say that

> for a judgement plan, this is reasonable, since although the hearer might not agree with the suggestion of error, he should realize that the referring expression [the domain plan] must be mutually acceptable in order for the identification to take place. For a refashioning, this also is reasonable, for if he doesn't find the resulting referring expression adequate, he can still accept it and then proceed to refashion it. (Heeman and Hirst, 1992, p. 19)

The ramification of this decision is that these moves need not be explicitly accepted and, therefore, can be automatically mutually accepted. In our model, this amounts to having an inference rule that allows an agent to infer the mutual belief that a plan of this type achieves its goal if he believes that it achieves its goal. Then, an agent can apply the mutual acceptance rule immediately without waiting for the other agent to accept the plan.

6.4.4 The Resulting Discourse Structure

The previous discussion has been about the acceptance process for one contribution, but in a discourse there are many contributions. We use the subplan in focus to be the current contribution that is to be reasoned about. This means, as described in section 6.2.2, that our focus stack encodes the intentional structure of a discourse. It specifies which contribution is currently being processed and which contribution needs to be mutually accepted before proceeding. Once the subplan in focus is mutually accepted, it is popped from the stack, the beliefs associated with it revise the mental state, and focus is shifted to another subplan (the next subplan in the stack). The resulting discourse structure is similar to that observed by Clark and Schaefer in real dialogues between humans.

The major difference between Clark and Schaefer's and our discourse structure is a consequence of not using their theory of evidence in expressing acceptance. In our model, an agent always explicitly accepts a plan by using the surface speech act `s-accept` (except

[4] They would have to be considered again if they were rendered invalid by the expansion, but in our model this is not possible. The effects in disjoint subplans are independent from each other.

in the case of judgement, refashioning, and suggestion plans, which are implicitly accepted). This means that the decision of whether or not to explicitly verbalize an acceptance is pushed to the speech generator, but, surely, this decision involves some knowledge available only at the level of the discourse planner. Giving this decision to the generator also means that the parser must be able to determine when an acceptance has been expressed, whether or not it was verbalized. This task is difficult when based solely on the syntactic and prosodic knowledge of a parser. Determining when an acceptance move has been made depends on the previous discourse as recorded in the focus stack.

A better, more comprehensive model would include reasoning rules to determine how much evidence should be given, and a set of acceptance plans that would allow for the various ways of expressing acceptance (which are not necessarily linguistic). Like all the plan schemas defined in this thesis, the acceptance plans would be applicable to both generation and understanding. For most plans, continued attention, or the initiation of the next relevant contribution would be enough, and these moves are equivalent to not verbalizing anything.

As noted earlier, the focus stack imposes a rigid structure onto the discourse. It is a model of the most prevalent type of structure seen in dialogues where information is being described, as in direction-giving dialogues (Clark and Wilkes-Gibbs, 1986; Clark and Schaefer, 1989; Psathas, 1991). We are able to model a large portion of real dialogue in a plausible way. The focus stack accounts for hierarchical acceptance structures, such as clarification subdialogues and other types of side sequences, and for hierarchical presentation structures, such as installment structures. But other types of structure are sometimes observed (see Clark and Schaefer (1989)), such as the the consideration of contributions top down instead of bottom up. In Chu's plan evaluator (Chu, 1993), the actions are examined in a top-down fashion so that the most general action that is inappropriate is addressed first. This ordering seems implausible because, if a general contribution is found unacceptable, the cause is probably an unacceptable sub-contribution.

Nevertheless, we feel that our model is a successful amalgamation of speech act theory, Clark and Schaefer's theory of contributions, and Grosz and Sidner's theory of discourse structure.

6.5 The Reasoning Process

The reasoning rules, along with an agent's other knowledge, are used during the reasoning process. The process actually consists of two separate subprocesses; one for understanding utterances, and one for generating a response. To take part in a conversation, the system runs an alternating sequence of the two subprocesses. The basic algorithms for the two subprocesses are shown in figures 6.3 and 6.2, respectively, and are described in greater

```
while their is input (surface speech actions)
    infer a plan from the input actions
    reason about the plan (i.e., adopt beliefs about its validity, etc.)
    while a mutual acceptance or collaborative activity rule can be applied
        apply the rule
```

Figure 6.2: Algorithm for understanding

```
while a goal adoption rule can be applied
    adopt the goal
    construct a plan that achieves the goal
    output the surface speech actions
    reason about the plan (i.e., adopt beliefs about the plan)
    while a mutual acceptance or collaborative activity rule can be applied
        apply the rule
```

Figure 6.3: Algorithm for responding

detail below.

To understand an utterance, the system acts as the hearer and infers a plan for each set of actions that it hears in turn. For each plan, the system determines whether the plan itself achieves its goal and whether its subplans (if any) achieve their goals (these plans and subplans are pushed to the focus stack). For each subplan, either the belief that it achieves its goal, or the belief that it is in error is added to the system's mental state. Then, the system applies any mutual acceptance rule or collaborative activity rule that it can. After processing each set of observed actions, the system switches roles and becomes the speaker.

To generate a response, the system, acting as speaker, attempts to apply any goal adoption rule. Once a goal is adopted, the system constructs a plan to achieve this goal, and adds the belief that the plan achieves its goal and the beliefs that any subplans achieve their goals (the system pushes the subplans to the focus stack). Then, presupposing the user's acceptance of the plan, it applies any mutual acceptance rule or collaborative activity rule to update its mental state. The system repeats this process until it can find no more goals to adopt, at which point it switches roles to become the hearer. There are two exceptions to always adopting a goal when it is possible to do so. First, a goal is not adopted to accept a plan in the same response that presented that plan, or an expansion to the plan. Second, a goal is not adopted to expand a plan from a suggestion in the same response as the suggestion (i.e., you do not expand a plan from your own suggestion). Planning for these two goals is pre-empted.

6.6 Summary

In this chapter we have completed our plan-based model of collaborative discourse. We have merged the work of Clark and Schaefer, Heeman and Hirst, and Grosz and Sidner to form a model capable of accounting for general task-oriented collaborative discourse.

The main feature of the model is the collaborative state that includes the current domain plan, a focus of attention, and the intention to achieve the goal of the domain plan. The current plan is in the common ground of the agents and they have the intention to keep it there. Hence, the collaborative state sanctions the adoption of goals that operate on the current plan, and sanctions the acceptance of these plans and of the domain plan (and its subplans).

The acceptance process for a contribution causes the agents to act collectively to update their common ground in an orderly way. Each contribution corresponds to an intention to achieve some goal or subgoal in the discourse. The focus stack maintains a structure of contributions similar to Grosz and Sidner's intentional structure, and specifies which contribution is being operated on by the current utterance.

The resulting discourse structure is plausible, because it is observed in many actual dialogues between humans. A few points to consider are the following. A record of the discourse is not kept. Only the current domain plan is kept in the agents' common ground and the discourse actions operate on it, resulting in an iterative structure. Older domain plans can be considered forgotten, or out of the scope of the discourse. The focus stack encodes the current intentional structure of the discourse, but past intentions are not remembered. The stack evolves over time, but at the end of the discourse segment (once the domain plan is mutually accepted) the stack is empty. A discourse progresses over time, but is not planned ahead of time. There is no way to even represent an overall discourse plan.

Although we concentrated on domain plans involving communicative action, specifically, route description plans, we believe that the model is applicable to other, more classical, domain tasks. Obviously, more reasoning rules might be necessary, and the discourse actions would need to be more general. We are unsure how the model could handle complex domain plans with interrelated actions, collective actions, and/or unintentional side-effects.

As noted previously, we have a very simple and inadequate account for how the acceptance of a plan is actually expressed. Because we are considering only linguistic actions, a plan must be accepted by an explicit use of an `s-accept` action. Therefore, we must assume that the generator and parser can realize and recognize acceptance moves. If it was possible to plan for other types of action, then a complete model of acceptance could be added to our current model.

In the next chapter, we present a full example of collaboration on direction-giving that should clarify how we have implemented our model and how it functions in our application domain.

Chapter 7

Example

This chapter presents a complete example of our system in action. In the example, we use dialogue (7.1) to illustrate how our system collaborates to make a referring action successful within the context of direction-giving.

(7.1)	1	G.	Go to the Lowell Street intersection.
	2	R.	Does it have a sign?
	3	G.	Yes, it does, and it also has traffic lights.
	4	R.	Okay.

Dialogue (7.1) is a simplified version of a portion of dialogue (1.1), an actual telephone conversation between humans, repeated below as dialogue (7.2).

(7.2)	1	A.	Can you tell me where the Academy is?
	2	B.	Yeah, where ya coming from?
	3	A.	uh Newton.
	4	B.	Okay, why dontcha come up 128?
	5	A.	Yes.
	6	B.	And take 2A.
	7	A.	Yes,
	8	B.	um 2A will take ya right across Mass Avenoo an ya just stay on 2A, **uh until ya get to Lowell Street.**
	9	A.	**Is it marked?**
	10	B.	**uh, Lowell Street?**
	11	A.	**Yeah.**
	12	B.	**Yeah I think there's a street sign there,** **its an intersection with lights.**
	13	A.	**Okay.**
	14	B.	an ya turn right on Lowell Street. an its about quarter to half a mile um, take another right on Bartlett Avenoo.
	15	A.	Okay.

16 B. an that takes ya right to the Academy.
17 A. Okay.

The simplification is necessary because dialogue (7.2) has many complexities that we have not addressed in this thesis. First, the directions given in the dialogue are given in a series of installments. Since we do not have an adequate model of how a route description plan (or, for that matter, any plan) can be communicated in installments, we restricted our example to the portion of dialogue (7.2) shown in boldface. Although our system can construct an entire route description plan (to be realized in one utterance), we chose this single installment to keep the example simple and plausible. Our main goal is to model the collaborative dialogue surrounding a reference in the context of direction-giving; so the rest of the route description is irrelevant to showing how the system collaborates with the user to make an inadequate referring plan adequate. (See section 3.5.1 for further discussion on the complexity of modelling installments.)

Within this installment there are further complexities. Agent B finds the utterance *Is it marked?* ambiguous and requests a clarification. We have not addressed this type of repair. Furthermore, we would need to model the attentional state of an agent in order to resolve and construct this and other anaphoric references in the dialogue.

Another complexity concerns world knowledge. The agents in this dialogue know that the reference *until ya get to Lowell Street* refers to an intersection and that for a street or an intersection to be marked is for it to have a sign. We have simplified the references that require inference from world knowledge.

Dialogue (7.1) is one installment of a route description, and contains an elaboration subdialogue. In the example, we consider this installment to be a single action in a sequence of direction actions that forms an entire route description plan. We will concentrate on the collaborative repair of the referring expression for the intersection. There are two agents participating in the dialogue, so, in the next two sections, we illustrate the dialogue from each agent's point of view. In the first, the system takes the role of the direction giver, and, in the second, the system takes the role of the direction recipient. Appendix B contains a trace of the system taking the role of the direction giver.

7.1 System as Direction Giver

Before the discourse commences, the system, acting as direction giver, has the following beliefs about the world, which correspond to the street network shown in figure 7.1:

```
category(highway1,highway).
called(highway1,'128').
category(street1,street).
called(street1,'2A').
```

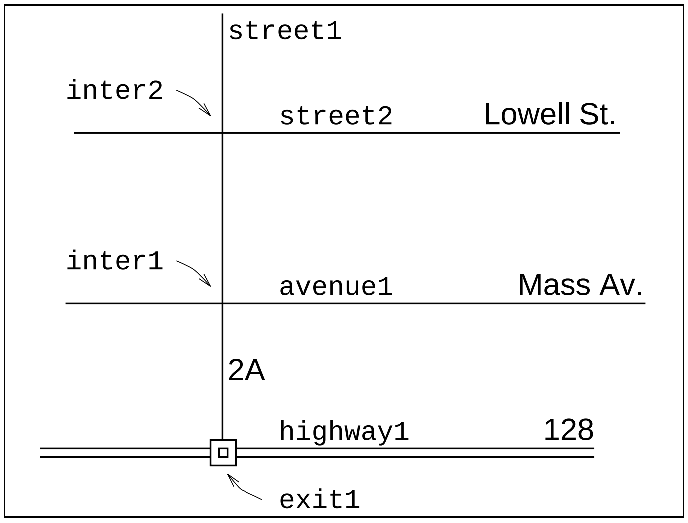

Figure 7.1: Street network known by the direction giver

```
category(avenue1,avenue).
called(avenue1,'Mass').
category(street2,street).
called(street2,'Lowell').

category(exit1,exit).
called(exit1,'2A').
category(inter1,intersection).
called(inter1,'Mass Avenue').
category(inter2,intersection).
called(inter2,'Lowell Street').
has(inter2,sign1).
has(inter2,lights1).
category(sign1,sign).
category(lights1,traffic-lights).

place(exit1, [highway1,street1],
         [(0,street1,+1), (90,highway1,-1), (270,highway1,+1)]).
place(inter1, [street1,avenue1],
         [(0,street1,+1), (90,avenue1,+1),
          (180,street1,-1), (270,avenue1,-1)] ).
place(inter2, [street1, street2],
         [(0,street1,+1), (90,street2,+1),
          (180,street1,-1), (270,street2,-1)] ).

path(highway1, [exit1]).
path(street1, [exit1,inter1,inter2]).
path(avenue1, [inter1]).
path(street2, [inter2]).
```

In addition, the system has the following beliefs about salience (these beliefs form only a subset of the system's two salience hierarchies):

```
salient-category(2,traffic-lights).
salient-category(1,intersection).
salient-category(1,sign).

salient-property(3,intersection,traffic-lights, λX·λY·has(X,Y)).
salient-property(2,intersection,sign, λX·λY·has(X,Y)).
salient-property(2,intersection, λX·called(X,Name)).

confidence-threshold(intersection,2).
```

7.1.1 Constructing "*Go to the Lowell Street intersection.*"

The system, acting as speaker, first adopts the initial goal to construct a route description plan for the entire route. Part of this plan is for describing the portion of the route originating at `exit1` and ending at `inter2`, which involves only one action: *going to the intersection.* After adopting the initial goal, the system constructs the route description plan shown in figure 7.2, where the installment that we are considering is shown. In the figure, arrows represent decomposition, and for brevity, constraints, mental actions, and effects are omitted, and only the parameters of surface speech actions are shown. Relevant instantiations are shown in a box at the bottom of the figure, and for some headers, their internal name (e.g., `p1`) is shown. In the rest of this example, we consider only those plans and subplans associated with this one installment.

To construct the direction action, the system used the `goto` action schema, which makes a reference to `inter2`, the place to go to. The system constructed a `refer` plan for `inter2` choosing the head noun, *intersection*, from the category of `inter2`. To disambiguate `inter2` from `inter1`, the system chose the simplest modifier, `called(inter2,'Lowell Street')`. The confidence value of the referring plan is 3 (1, for the category, plus 2, for the name), which exceeds the system's threshold of 2.

The system reasons about the plan and pushes it and its four subplans onto the focus stack, shown below with plan names (e.g., `p1`) representing actual plan derivations. Note the order in which the subplans are pushed: top to bottom, left to right.

```
(p1,p44,mb(system,user,called(inter2,'Lowell Street')))
(p1,p29,mb(system,user,category(inter2,intersection)))
(p1,p20,identify(user,system,entity(1,inter2)))
(p1,p13,mb(system,user,place(inter2,_,_)))
(p1,p1,knowroute(user,system,
           route([...,goto(exit1,inter2,street1,+1,1),...])))
```

For each plan on the stack, the system believes that it achieves its goal; so, the system adds

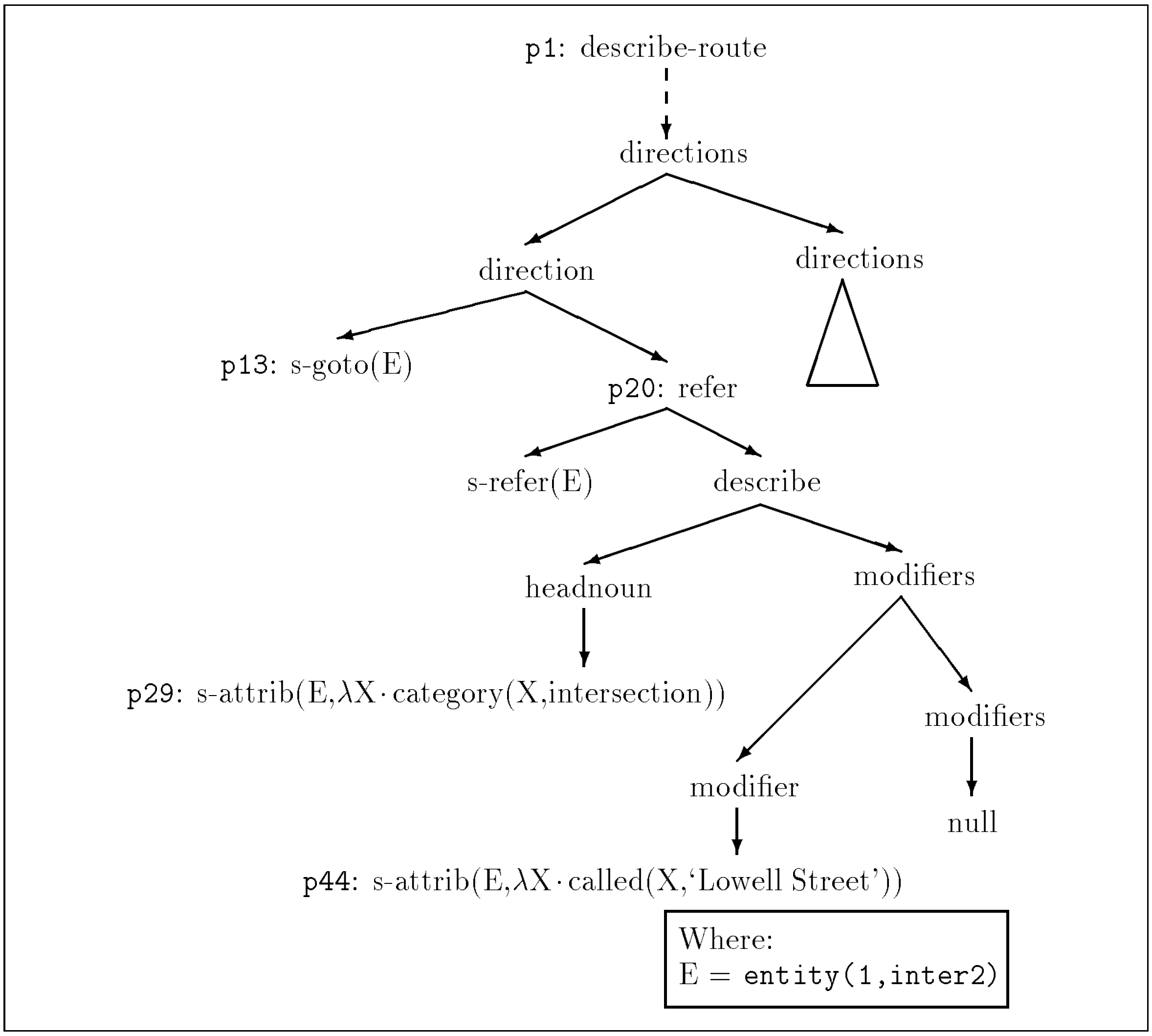

Figure 7.2: Route description plan derivation showing subplan for *Go to the Lowell Street intersection.*

the following beliefs to its mental state:

```
achieve(p1,p44,mb(system,user,called(inter2,'Lowell Street')))
achieve(p1,p29,mb(system,user,category(inter2,intersection)))
achieve(p1,p20,identify(user,system,entity(1,inter2)))
achieve(p1,p13,mb(system,user,place(inter2,_,_)))
achieve(p1,p1,knowroute(user,system,
            route([...,goto(exit1,inter2,street1,+1,1),...])))
```

The system outputs the following surface speech actions to the generator:

```
s-attrib(entity(1,inter2),λX·called(X,'Lowell Street')),
s-attrib(entity(1,inter2),λX·category(X,intersection)),
s-refer(entity(1,inter2)),
s-goto(entity(1,inter2)).
```

These could be realized as

Go to the Lowell Street intersection.

The system then attempts to apply any mutual acceptance rule or collaborative activity rule. The only rule that the system can apply is for entering into a collaborative activity. Applying this rule causes the system to adopt the following belief:

```
cstate(system,user,p1,knowroute(user,system,
          route([...,goto(exit1,inter2,street1,+1,1),...])))
```

The system is now in a collaborative state that will not end until the route description is mutually accepted. Finally, the system becomes the hearer, waiting for an utterance from the recipient.

7.1.2 Understanding "*Does it have a sign?*"

The recipient utters *Does it have a sign?* which has the following underlying surface speech actions:

```
s-accept(p1).
s-accept(p1).
s-postpone(p1).
s-suggest(p1,[s-attrib-rel(entity(1,inter2),entity(N,Obj),
                            λX·λY·has(X,Y)),
            s-refer(entity(N,Obj)),
            s-attrib(entity(N,Obj),λX·category(X,sign))]).
```

We assume that the parser can determine that these actions underlie the utterance. In particular, we make the tenuous assumption that the parser can figure out that two `s-accept` actions are necessary for the postponement and suggestion to make sense. The system processes each surface speech action in turn, and in the order shown.

The system performs plan inference with the first action as input, and comes up with an instance of `accept-plan` that expresses acceptance of the subplan in focus, `p44`. The system then reasons that the `accept-plan` is achieved because it believes that the subplan in focus achieves its goal, and so shifts focus to the `accept-plan`, by pushing it onto the stack. The next step is to apply the mutual acceptance rule as many times as is possible. The system applies it to mutually accept the `accept-plan` which allows the system to adopt the belief in the mutual belief that `p44` achieves its goal (i.e., the goal of the mutual belief in the name of the intersection):

```
mb(system,user,achieve(p1,p44,
          mb(system,user,called(inter2,'Lowell Street')))).
```

This application pops the accept-plan from the stack. The system applies the rule a second time to mutually accept the s-attrib action, which allows the system to believe the mutual belief that the intersection is called *Lowell Street*:

```
mb(system,user,called(inter2,'Lowell Street')).
```

The subplan, p44, is popped from the stack.

Next, the system performs plan inference with the second s-accept action as input. The process is similar to the former acceptance, resulting in the system believing that:

```
mb(system,user,achieve(p1,p29,
          mb(system,user,category(inter2,intersection)))).
mb(system,user,category(inter2,intersection)).
```

The third action, s-postpone(p1), is given to the plan inference process, which infers an instance of postpone-plan. The goal of this action is to establish the mutual belief that the subplan in focus, p20, has an error, but the system does not immediately believe this mutual belief. First, the system shifts focus to the postponement plan, which the system believes is implicitly achieved, and therefore mutually acceptable. This application of the mutual acceptance rule allows the system to believe the mutual belief that the refer plan, p20, has an error:

```
mb(system,user,error(p1,p20,p55)).
```

The evaluation of the postponement plan determined that the error was at node p55, whose parent action (i.e., the final modifiers action in the plan) has the confidence-exceed constraint. Adopting this belief causes the system to retract the belief that p20 achieves its goal. Focus has shifted back to p20, but p20 is not popped from the stack, because it has not been mutually accepted yet.

A second effect of the postponement action is that the system raises its confidence threshold, because of the underlying intention of the user to inform the system that she is not confident in the adequacy of the referring plan. The new confidence threshold is the following:[1]

```
confidence-threshold(intersection,6).
```

The fourth and final surface speech action, after being inferred from, results in an instance of suggest-expand-plan. Through a similar reasoning process to that for the postponement plan above, the system believes that it is mutually believed that:

[1]We have no theory about how much the threshold should change, other than that it should be raised by about the amount that one or two modifiers would raise the overall confidence value of a referring plan. As we will see below, different values will result in different elaborations.

```
suggest(p1,[s-attrib-rel(entity(1,inter2),entity(N,Obj),
                          λX·λY·has(X,Y)),
            s-refer(entity(N,Obj)),
            s-attrib(entity(N,Obj),λX·category(X,sign))]).
```

This is a suggestion to expand the plan with a modifier for having a sign.

Since there are no more surface speech actions to consider, the system switches to become the speaker again. The current state of the focus stack is the following:

```
(p1,p20,identify(user,system,entity(1,inter2)))
(p1,p13,mb(system,user,place(inter2,_,_)))
(p1,p1,knowroute(user,system,
           route([...,goto(exit1,inter2,street1,+1,1),...])))
```

7.1.3 Constructing "*Yes, it does, and it also has traffic lights.*"

The system now attempts to apply any goal adoption rule that it can. Since the system believes there is an error in the referring plan, it adopts the goal to replace the current route description plan with a new plan. To achieve this goal, the system finds that it can use an expansion move, but there are two such moves. The first, to expand using only its own beliefs, cannot be used because the user made a suggestion. So the system constructs an instance of `expand-plan`, and attempts to incorporate the user's suggestion into the expansion.

The first step is to construct a `modifiers` schema that serves to elaborate on the previous description (*the Lowell Street intersection*), and so the plan constructor is called recursively. The system uses the suggestion because it believes the intersection does have a sign, but the confidence value of this modifier is only 2. This value brings the overall confidence value of the referring expression up to 5, which does not exceed the threshold of 6. So another modifier, that the intersection has traffic lights, is also added to the `modifiers` sequence. Now the overall confidence value is 8 (because the confidence value of an intersection having traffic lights is 3), which exceeds the threshold, and, thus the recursive call to the constructor can terminate.

This expansion is then substituted for the step of the old plan that had the error in its constraint (node `p55`). That is, the final `modifiers` step with the `null` decomposition is replaced with the newly constructed modifiers sequence. As a result the new referring expression is *the Lowell Street intersection that has a sign and traffic lights*, whose plan is shown in figure 7.3.

However, the system does not utter the whole referring expression, because part of it was already uttered. The next step in constructing the `expand-plan` action is to decide how to respond to the suggestion. Since the suggested modifier and an extra modifier were used to elaborate the referring expression, the system chooses to affirm that the intersection

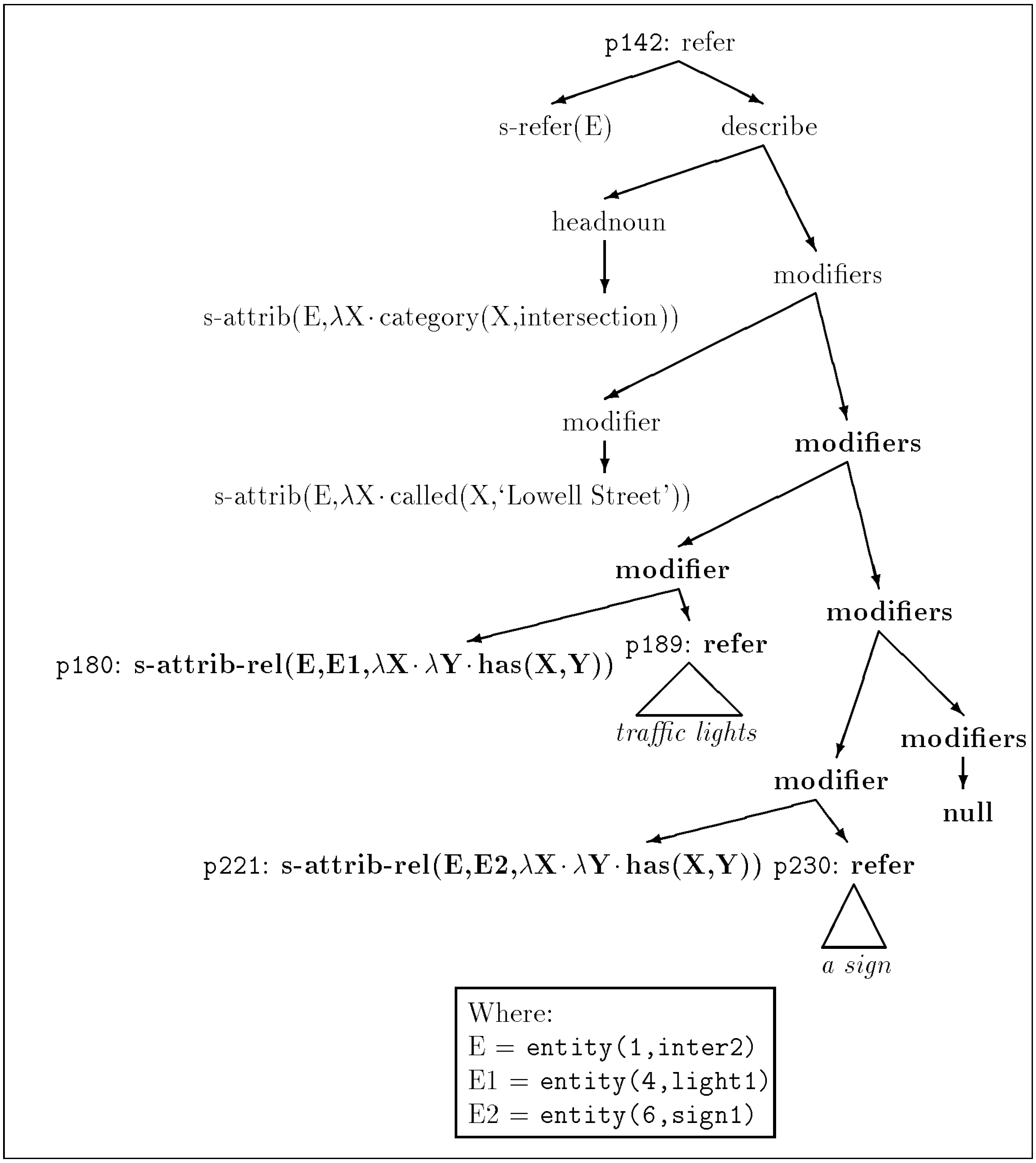

Figure 7.3: Referring plan derivation for *the Lowell Street intersection that has a sign and traffic lights.* (The expansion is shown in bold type.)

does have a sign (thus informing the user that the suggestion was used), and to inform the user of the extra modifier. The surface speech actions to be given to the generator are:

```
s-actions(p1,[s-attrib(entity(4,lights1),
                       λX·category(X,traffic-lights)),
              s-refer(entity(4,lights1)),
              s-attrib-rel(entity(1,inter2),entity(4,lights1),
                           λX·λY·has(X,Y))]).

s-affirm(p1,[s-attrib-rel(entity(1,inter2),entity(6,sign1),
                          λX·λY·has(X,Y)),
             s-refer(entity(6,sign1)),
             s-attrib(entity(6,sign1),λX·category(X,sign))]).
```

which could be realized as

Yes, it does, and it also has traffic lights.

The system could have responded in two other ways depending on the constructed expansion. First, if the confidence threshold had been adjusted to 4 instead of 6, then the use of the suggestion would have been enough to make the plan adequate, and the following surface speech action would be given to the generator:

```
s-affirm(p1,[s-attrib-rel(entity(1,inter2),entity(6,sign1),
                          λX·λY·has(X,Y)),
             s-refer(entity(6,sign1)),
             s-attrib(entity(6,sign1),λX·category(X,sign))])
```

which could be realized as *Yes, it does have a sign.*

Second, if the system did not believe that the intersection had a sign, then it would not be possible to use the suggestion. The system would have to find a different way to expand the plan. Assuming that adding the modifier that the intersection has traffic lights makes the referring plan adequate, the following two surface speech actions could be realized as *No, it doesn't have a sign, but it has traffic lights.*

```
s-actions(p1,[s-attrib(entity(4,lights1),
                       λX·category(X,traffic-lights)),
              s-refer(entity(4,lights1)),
              s-attrib-rel(entity(1,inter2),entity(4,lights1),
                           λX·λY·has(X,Y))]).

s-deny(p1,[s-attrib-rel(entity(1,inter2),entity(6,sign1),
                        λX·λY·has(X,Y)),
           s-refer(entity(6,sign1)),
           s-attrib(entity(6,sign1),λX·category(X,sign))]).
```

Returning to the `expand-plan` that was constructed, the next move is to reason about it. Since refashioning plans are implicitly achieved, the system adds this belief to its mental state, and pushes the plan to its focus stack.

Now the system attempts to apply the mutual acceptance rule as many times as is possible, because it presupposes that the user will accept the plan it has just constructed. The `expand-plan` gets automatically mutually accepted, because the system can infer the mutual belief that it is achieved from its own belief. This allows the system to update its mental state with the mutual belief that the new elaborated plan replaces the old plan, which causes belief revision.

To revise its belief about the current route description plan the system does several things.

First, it removes the old plan from its mental state, and adds the new plan.

Second, the system updates its focus stack, by replacing the plan names and subplan names that were part of the old plan with their equivalent names in the new plan. Since these plans are on the focus stack, they have not been mutually accepted yet, but the system's beliefs about their validities may have changed as a result of the new expansion, so the system updates these beliefs too. Since the system evaluated the new plan during its construction, the system believes that all of the new subplans achieve their goals.

Third, the system determines which subplans of the plan are new, when compared with the old plan, pushes these to its focus stack, and adds beliefs that they achieve their goals. The new focus stack contains the following elements:

```
(p123,p239,mb(system,user,category(sign1,sign)))
(p123,p230,identify(user,system,entity(6,sign1)))
(p123,p221,mb(system,user,has(inter2,sign1)))
(p123,p198,mb(system,user,category(lights1,traffic-lights)))
(p123,p189,identify(user,system,entity(4,lights1)))
(p123,p180,mb(system,user,has(inter2,lights1)))
(p123,p142,identify(user,system,entity(1,inter2)))
(p123,p135,mb(system,user,place(inter2,_,_))
(p123,p123,knowroute(user,system,
           route([...,goto(exit1,inter2,street1,+1,1),...])))
```

Note that the new plan name is `p123` and that the three subplans that used to be on the stack are still on the stack, but at the bottom.

Finally, the system updates its belief about the collaborative state by replacing the old plan name, `p1`, in its `cstate` belief with `p123`.

The system cannot apply the mutual acceptance rule again, so tries to adopt another goal. The only possible goal is to establish the mutual belief that the subplan in focus (at the top of the stack above) achieves its goal. The system pre-empts its construction of an `accept-plan` because it just constructed the subplan in focus itself, and must wait until the user accepts the subplan.

With nothing left to do, the system becomes the hearer once again, and waits, hopefully, for the user's acceptance of its new plan.

7.1.4 Understanding "*Okay.*"

The user does indeed accept the new plan by uttering *Okay*, whose underlying set of surface speech actions consists of *eight* instances of

```
s-accept(p123).
```

We assume that the parser can determine that this utterance is intended to accept the whole plan so far which contains the eight subplans on the focus stack.[2]

For each of these actions, the system infers an instance of `accept-plan`. The acceptance process that follows is identical to that described above for the two `s-accept(p1)` surface speech actions. The top eight subplans on the focus stack are eventually popped from the stack, and for each subplan, its goal is believed to be true. Only one plan remains on the focus stack:

```
(p123,p123,knowroute(user,system,
            route([...,goto(exit1,inter2,street1,+1,1),...])))
```

and as discussed in footnote 2, this plan would also be removed from the stack if the user has accepted the route description. This event would occur after the last direction (or installment) was mutually accepted, and if the user believed that route description plan was complete.

Once the route description is mutually accepted, the focus stack would be empty, and thus the collaborative activity of giving the directions would terminate, leaving the participants to return to bigger and bolder topics.

7.2 System as Direction Recipient

Switching point of view, we re-examine the dialogue with the system acting as the direction recipient. The system has the following beliefs about the world:

```
category(exit1,exit).
called(exit1,'2A').
category(street1,street).
```

[2] In this example we have been considering only one installment of the route description. In a complete dialogue, if this installment was the last (or only) installment, then the user would also express her judgement of the entire route description plan; or if there were more directions following this installment, then the user would judge these directions, or would wait for them to be given. In the example, we have omitted any surface speech actions pertaining to the dialogue that could occur after the installment.

```
called(street1,'2A').

place(exit1, [street1], [(0,street1,+1)]).
path(street1, [exit1]).
```

The system has knowledge of the origin, but not of any other location on the route. We have omitted the system's belief about the ultimate destination of the route, because it is not relevant to the installment we are considering.

In addition, the system has the following beliefs about salience:

```
salient-category(2,traffic-lights).
salient-category(1,intersection).
salient-category(1,sign).

salient-property(3,intersection,sign, λX·λY·has(X,Y)).
salient-property(2,intersection, λX·called(X,Name)).
salient-property(1,intersection,traffic-lights, λX·λY·has(X,Y)).

confidence-threshold(intersection,5).
```

Although the system's category salience hierarchy is the same as that of the direction giver, its property salience hierarchy is different. Also, the confidence threshold of the recipient is higher than the giver's.

7.2.1 Understanding "*Go to the Lowell Street intersection.*"

The system is given as input the surface speech acts underlying *Go to the Lowell Street intersection* as shown below:

```
s-attrib(entity(1,Obj),λX·called(X,'Lowell Street')),
s-attrib(entity(1,Obj),λX·category(X,intersection)),
s-refer(entity(1,Obj)),
s-goto(entity(1,Obj)).
```

The system performs plan inference, which first finds a plan derivation with this set of surface speech actions as its yield. This results in the plan derivation shown in figure 7.4 that is almost identical to the derivation in figure 7.2. The only difference is that none of the parameters are instantiated, and, in particular, the discourse entity, E, of the reference is uninstantiated.

Next, the system evaluates the plan derivation. A new object name, `object1`, is created for the new reference, and a new discourse entity, `entity(1,object1)`, is instantiated. Evaluating the direction portion of the plan results in the system adding the belief to its own beliefs that the giver believes that `object1` is a place:

```
bel(user,place(object1,_,_)).
```

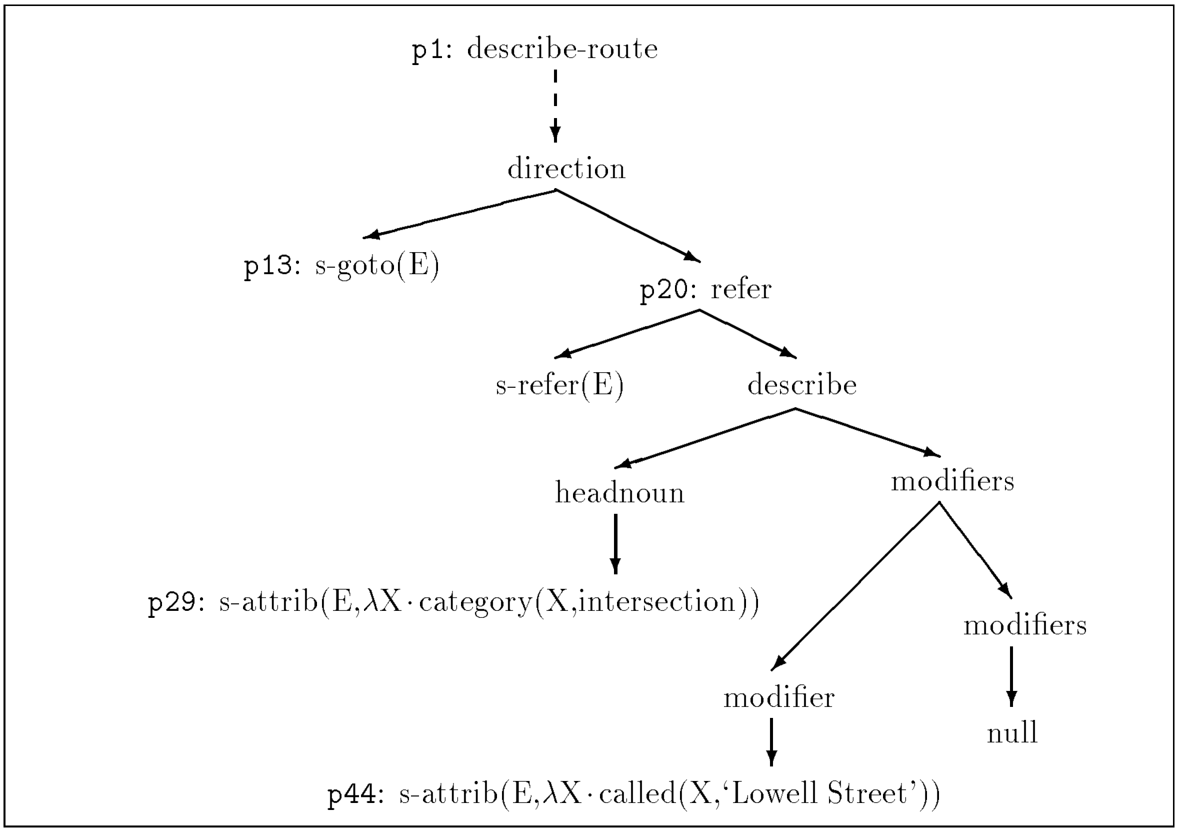

Figure 7.4: Inferred plan derivation for *Go to the Lowell Street intersection.*

The system then evaluates the `refer` plan, `p20`, as shown in the example in section 4.5.2. As a result the system adds the following beliefs to its mental state:

```
bel(user,category(object1,intersection)).
bel(user,called(object1,'Lowell Street')).
```

Also, the final confidence value of the referring expression plan is 3, which does not exceed the system's confidence threshold of 5, so the constraint (node `p56`, which is not shown) in the final modifiers plan schema fails.

Now that plan inference is finished, the system reasons about the inferred plan. For each subplan, the system pushes it onto the focus stack, and decides whether or not it achieves its goal. Here, the first two subplans, `p44` and `p29`, achieve their goals because the system believes that the user believes the properties (and also believes that the user is rational). However, the `refer` plan itself, `p20`, does not achieve its goal, because the system is not confident in its adequacy. So, the system adds the plan derivation to its mental state along with the following beliefs:

```
achieve(p1,p44,mb(system,user,called(object1,'Lowell Street'))).
achieve(p1,p29,mb(system,user,category(object1,intersection))).
error(p1,p20,p56).
```

The resulting focus stack is as follows:

```
(p1,p44,mb(system,user,called(object1,'Lowell Street')))
(p1,p29,mb(system,user,category(object1,intersection)))
(p1,p20,identify(user,system,entity(1,object1)))
(p1,p13,mb(system,user,place(object1,_,_)))
(p1,p1,knowroute(user,system,
           route([...,goto(exit1,object1,street1,+1,1),...])))
```

The final step of understanding the speaker's utterance is to apply the collaborative activity rule to enter into the collaborative state.

7.2.2 Constructing "*Does it have a sign?*"

The system switches from being hearer to being speaker and attempts to apply any goal adoption rule that it can. Since the system believes that the subplan in focus, `p44`, achieves its goal, it adopts the goal of making this belief mutually believed. The system then constructs an instance of `accept-plan` that accepts `p44`, and since the system believes that `p44` achieves its goal, it can adopt the belief that the `accept-plan` achieves its goal (and can also pushes the `accept-plan` onto the focus stack).

The system then applies the mutual acceptance rule twice, presupposing the user's acceptance of the judgement plan. The first time pops the `accept-plan` from the stack, establishing the mutual belief that `p44` achieves its goal, and the second time pops `p44` from the stack, establishing the mutual belief that the referent is called *Lowell Street*.

The system repeats this same cycle again to accept the next subplan, `p29`, on the focus stack, which establishes the mutual belief that the referent is an intersection.

Now, the subplan in focus is the `refer` plan, `p20`. Since the system believes this subplan is invalid, it adopts the goal of making the invalidity mutually believed, and therefore constructs an instance of `postpone-plan`. Next, through a reasoning process similar to those used in the previous two acceptance moves, the system establishes the mutual belief that the subplan has an error.

Next, the system checks to see if any other goal can be adopted. It attempts to expand the plan to make it adequate, but this is not possible, because the system has no knowledge of the referent. Instead, it adopts the goal of making a suggestion, and so constructs an instance of `suggest-expand-plan` (as shown in the example in section 5.5.1), which it pushes onto the focus stack. In the suggestion, the system asks if the intersection has a sign, because the property of having a sign is the most salient property for an intersection that has not already been used in the referring expression. Presupposing the user's acceptance of the suggestion, the system applies the mutual acceptance rule, which establishes the mutual belief that a suggestion has been made.

Finally, the system attempts to adopt another goal. The only possible goal to adopt is the goal of replacing the current plan with a new plan (via an `expand-plan`), but the system pre-empts the plan construction process, because it has just made a suggestion.

The four plans that the system has constructed give rise to the following surface speech actions:

```
s-accept(p1).
s-accept(p1).
s-postpone(p1).
s-suggest(p1,[s-attrib-rel(entity(1,object1),entity(N,Obj),
                           λX·λY·has(X,Y)),
              s-refer(entity(N,Obj)),
              s-attrib(entity(N,Obj),λX·category(X,sign))]).
```

which could be realized collectively as

Does it have a sign?

Finally, the current state of the focus stack is the following:

```
(p1,p20,identify(user,system,entity(1,object1)))
(p1,p13,mb(system,user,place(object1,_,_)))
(p1,p1,knowroute(user,system,
            route([...,goto(exit1,object1,street1,+1,1),...])))
```

7.2.3 Understanding "*Yes, it does, and it also has traffic lights.*"

The user's reply of *Yes, it does, and it also has traffic lights* has the following underlying surface speech actions, which are input to the system:

```
s-affirm(p1,[s-attrib-rel(entity(1,Obj),entity(3,Obj1),λX·λY·has(X,Y)),
             s-refer(entity(3,Obj1)),
             s-attrib(entity(3,Obj1),λX·category(X,sign))]).

s-actions(p1,[s-attrib-rel(entity(1,Obj2),entity(4,Obj2),
                           λX·λY·has(X,Y))])
              s-refer(entity(4,Obj2)),
              s-attrib(entity(4,Obj2),λX·category(X,traffic-lights)).
```

Once again, the system's first task is to perform plan inference. This results in an `expand-plan` plan derivation that has the goal of replacing the old route description plan with a new expanded plan. During the evaluation, a `modifiers` action is found that has as its yield the speech actions in the `s-affirm` and `s-actions` above. This plan is then substituted for the invalid `modifiers` action in the old plan. The resulting plan derivation contains a referring expression plan similar to the one shown in figure 7.3, except that the discourse entities are not instantiated. The evaluation finds the `expand-plan` to be valid, so, after pushing it onto the focus stack, the system adopts the belief that it achieves its goal.

Since an expansion plan implicitly achieves its goal, the system can apply the mutual acceptance rule, which allows it to adopt the mutual belief that the new route description plan replaces the old plan. To adopt this belief, the system has to revise its beliefs about the current plan of the collaborative activity, and so must make the new plan the current plan. The first step in the revision is to evaluate the new plan. The evaluation results in the creation of two new object names, `object2` and `object3`, corresponding to the two new referents in the plan, and these are used to instantiate the discourse entities in the plan. The system also adopts the following beliefs about the properties of the referents:

```
bel(user,has(object1,object2))
bel(user,category(object2,sign))
bel(user,has(object1,object3))
bel(user,category(object3,traffic-lights))
```

The overall confidence value of the reference to the intersection has increased to 7 (original value of 3, plus 3, for having a sign, plus 1, for having traffic lights), which exceeds the system's confidence threshold of 5. Thus, the referring plan is found valid by the evaluation.

The second step of the revision is to reason about the new plan. This involves pushing the new subplans onto the focus stack, and adopting beliefs that each subplan achieves its goal. The resulting focus stack, shown below, contains the subplans that were already on the stack, and the new subplans that have not been mutually accepted yet. Note that the plan names have also been updated.

```
(p123,p239,mb(system,user,category(object2,sign)))
(p123,p230,identify(user,system,entity(6,object2)))
(p123,p221,mb(system,user,has(object1,object2)))
(p123,p198,mb(system,user,category(object3,traffic-lights)))
(p123,p189,identify(user,system,entity(4,object3)))
(p123,p180,mb(system,user,has(object1,object3)))
(p123,p142,identify(user,system,entity(1,object1)))
(p123,p135,mb(system,user,place(object1,_,_))
(p123,p123,knowroute(user,system,
            route([...,goto(exit1,object1,street1,+1,1),...])))
```

The final step of the revision is to update the current plan of the collaborative state with the new route description plan.

7.2.4 Constructing "*Okay.*"

The system becomes the speaker and expresses its acceptance of the eight subplans on the top of the stack. It cycles through the mutual acceptance process eight times in exactly the same way as described in section 7.2.2 above, and sends eight instances of

```
s-accept(p123).
```

to the generator, which could be realized collectively as *Okay.*

The installment that we have been considering has been mutually accepted, and, had it been the last installment (as discussed in footnote 2), the whole route description could have been accepted. But since this installment is not the last, the system considers the route description plan to be invalid, and so cannot accept the final plan on the focus stack:

```
(p123,p123,knowroute(user,system,
            route([...,goto(exit1,object1,street1,+1,1),...])))
```

If we had considered an entire route description plan, then, after accepting the plan, the system's focus stack would be empty, signalling the end of the collaborative activity.

Chapter 8

Conclusion

8.1 Contributions

We have presented a computational model of how two agents collaborate to make referring actions successful during direction-giving dialogues. This thesis makes several contributions to the field.

The central goal of this thesis was to model how an agent can refer to an object of which another agent has no knowledge in a collaborative environment. To this end, we suggested that a referring expression plan is acceptable to an agent if it contains a description that the agent can use to identify the referent eventually, and if the agent is confident that the plan is adequate for this purpose. We have assumed that, in our system, every description could be used as an identification plan. We developed a model of confidence based on the salience of the components of a referring expression that accounts for an agent's belief in the adequacy of the expression. We gave an algorithm for constructing the most salient referring expression according to an agent's beliefs. We have also provided a set of discourse actions that agents can use to express judgements, make elaborations, and suggest elaborations, so that the agents can collaborate to make the referring action successful.

Our model of salience can be applied to other types of referring action. It should always be important to make salient referring expressions. Likewise, our discourse actions for making suggestions, and elaborating referring expressions are applicable to other types of referring.

The model is not limited to reference in direction-giving dialogues. We view a referring expression plan as a means of identifying a referent, once the agent establishes the relevant context (which, in direction-giving, occurs as an agent traverses a route). Reference of this sort can be seen in any dialogue where an agent is referring to something new that is not visible or not known to exist by the other agent. Although less common than other types of reference, it is pervasive in, at least, instructional dialogues, if not all discourse.

A second goal was to augment Heeman and Hirst's model of collaborative discourse to handle larger domain plans with many subplans, subgoals, and effects. The collaborative state sanctions the adoption of goals to clarify or elaborate the plan (or subplan) in focus, and sanctions the mutual acceptance of the plan in focus. Focus of attention is maintained as a stack of plans (and subplans) that have not yet been mutually accepted. The stack results in a plausible discourse structure that allows clarification subdialogues as seen in real dialogues between humans. The focus stack imposes a rigid structure on a discourse, but it allows large domain plans to be negotiated in an orderly way. Our hope is that this collaborative discourse model is general enough to be applicable in domains other than direction-giving, and in domains that do not directly involve communicative action. However, because of the rigid structure, problems may arise if the domain plans have complex dependencies between actions, collective actions, interleaved actions, etc.

A third contribution lies in the area of interactive direction-giving. We have developed a preliminary model of direction-giving within a plan-based formalism that handles route description as a goal-oriented communicative task. The model is applicable to direction-giving discourse, which is necessary for interactive route description, route advice, route justification, and, perhaps, interactive route planning. The model is centred on the description of places and paths, because these descriptions are the most important, but could be easily extended to describe the actions that must be taken along a route.

8.2 Assumptions

We have made a few assumptions that are worth noting. First, we have assumed that the agents have mutual knowledge of the following items:

- the plan library (i.e., the set of available action schemas),
- the plan construction, inference, and reasoning processes,
- the acceptance process, including the use of the focus stack, and the collaborative state (we assume the agents know how to collaborate), and
- the mental actions for determining the reference context, computing confidence, and determining salience.

This knowledge can be considered part of the social contract that agents are accountable to when participating in social tasks.

Second, we have assumed that the parser and language generator are capable of recognizing and using the the surface speech acts appropriately. This is not a simple task, but we have tried to define surface speech actions so that their translation to and from utterances is plausible. However, we had to make the tenuous assumption that a parser can determine an acceptance move without access to the current context, or to the current domain plan.

Third, in relation to direction-giving, since we have considered route description in isolation from any other associated task, we have assumed that a route planner provides input to our system as a sequence of edges and nodes (as a path in a graph structure), or as a sequence of *goto* and *turn* actions. We are not addressing route planning in this thesis, only route description. But, because we have split these two, mostly independent tasks, we cannot handle the repair of an action that is too complex. This would require going back to the route planner, and replanning a segment of a route. Additionally, we cannot handle other phenomena seen in direction-giving, such as mistake detection and recovery.

8.3 Comparisons to Similar Work

In this section we compare our research to similar work. We first look at research into referring expressions, and, second, at research into collaboration.

Referring Expressions

Heeman and Hirst's model (Heeman and Hirst, 1992) and Clark and Wilkes-Gibbs's model (Clark and Wilkes-Gibbs, 1986) highly influenced our model of reference. The main difference is the type of referring action and its underlying intentions, which results in a different method of evaluating referring expressions and a different method of refashioning them. In their model, a referring expression can be overconstrained or underconstrained, and is refashioned by making a clarification; whereas, in our model, a referring expression can be inadequate, requiring elaboration. Nevertheless, the same acceptance cycle can be applied in both models, and the discourse actions are very similar.

Reiter and Dale's theory (Reiter and Dale, 1992) of generating referring expressions involves an algorithm that uniquely specifies a referent while choosing the most preferred attributes of the referent. Preference is given to visually salient attributes, such as size, shape, and colour, but salience does not depend on the context, or even on the referent's type. Our algorithm is similar, because it also disambiguates the referent from all others in its context set, but we allow salience to depend on the category of the referent, which results in better, more plausible, referring expressions. Reiter and Dale's theory only concerns the generation of referring expressions, and is not agent-oriented (i.e., they are not concerned with the intentions, goals, and beliefs of an agent making a reference). By considering referring in a collaborative and interactive context, we provide a motive for the underlying intentions of the referring act.

Appelt (1985c) proposed a type of referring, called *nonshared concept activation with identification intention*, for which he never completed a theory. We picked up where he left off, but modified his proposal in two ways. First, his *identification intention* includes both locating and recognizing the referent, whereas ours includes only the recognition part. Thus, to identify a referent, an agent has to be in the correct location, and this location should be known from the context of the reference. Second, he intended the referring expression to be used as the basis for an identification plan that could be executed to identify the referent. We take the referring expression plan itself to be the identification plan.

Collaboration

Our collaborative model has been highly influenced by Heeman and Hirst's model. The problem with their model, which results from only accounting for referring expression plans, is that it could not handle large domain plans with multiple goals and effects. We have extended their model by adding a focus stack to the collaborative state that an agent is in. This allows an agent to attend to a specific subplan in a domain plan (or a discourse plan) that he can judge, refashion, or accept without being concerned about the rest of the plan. The focus stack structures the discourse, so, in effect, we have merged the work of Grosz and Sidner (1986) on intentional structure, Clark and Schaefer (1989) on the acceptance process, and Heeman and Hirst (1992) on collaboration and discourse planning.

Lochbaum, Grosz, and Sidner (Lochbaum, Grosz and Sidner, 1990; Grosz and Sidner, 1990) deal with collaboration on shared domain plans. They concentrate on domain plans in which the agents act collectively to achieve goals, and provide a taxonomy of action types. The main difficulty with their theory is that a shared plan contains only actions that have been mutually agreed to contribute to the goal of the plan. Their theory cannot account for two types of behaviour that ours can. First, an agent sometimes proposes an action that another agent, or even himself, does not believe contributes. Without being able to incorporate these beliefs into the plan, the model cannot represent the state in which an agent understands how an utterance could contribute, without believing that it does. Second, adding a new action to a shared plan may invalidate an action already in the plan. The invalidated action would then have to be addressed. This state can never occur in their model. In our model, the current domain plan can contain actions that are not valid, so an agent has beliefs about the validity of every subplan of the domain plan. Agents have, when in the collaborative state, the intention to achieve the goal of the domain plan, and so also have the intention to repair the errors in the plan.

Lambert and Carberry (1991) propose a tripartite model of collaborative discourse, consisting of a discourse layer, a problem-solving layer, and a domain layer. In our model, there is no division between different types of planning. The same processes function on all the plans. The distinction between separate layers is blurred when the domain task directly involves communicative action.

Chu (1993), in her extension of Lambert and Carberry's model, breaks the dialogue model into two components: the existing, mutually agreed upon, plan that is never re-evaluated, and the proposed additions to the plan. On the surface, this model is very similar to ours, because the agents can collaborate on the proposed actions without having to believe that they contribute to the plan. The problem is that the existing plan can only be added to, which means the model cannot represent how the existing plan could be modified to accommodate new proposals. Our model is simpler, because both valid and invalid actions are kept in one plan, which is operated on by one set of discourse actions.

8.4 Future Directions

This work can be extended in many directions. First, we summarize some of the specific extensions that we have already been discussed in the text, and second, we mention a few general directions for future research.

Specific Directions

A complete model of route description plans, and, indeed, any model of communicative action, should account for how the plans can be communicated in a sequence of installments. Using installments is conducive to collaborative behaviour, because it breaks up complex material making it easier to understand and assimilate. In chapter 3 we outlined an approach that could be implemented within our plan-based model.

Our model of direction-giving is not as extensive as some of the other cited models, because we have designed only two direction action schemas: one for *going to* a place, and a second for *turning* onto a path. Obviously, many more schemas would be necessary to cover the wide variety of instructions that people use. We believe that our model is a sufficient framework to build from, because Kuipers's topological structures can represent most of the necessary knowledge for traversing an environment, and because the plan-based model is versatile enough to account for many types of action, including non-linguistic actions.

An agent computes his confidence in the adequacy of a referring expression by adding up the various numeric confidence values of the modifiers in the expression. As noted earlier, a confidence 'algebra' may be necessary since the confidence values of the modifiers might not be independent. They may interact with each other to raise or lower confidence. Also, the confidence in sub-expressions of the main referring expression should be taken into account.

To determine the most salient objects at a location, the direction giver, who has extensive knowledge of the location, resorts to accessing a hierarchy of salient features indexed by object type. Ideally, since salience is highly context dependent, the giver should apply an algorithm to determine what objects, and what properties, are the most salient, given the context.

General Directions

Perhaps the most obvious way to extend this work would be to merge the different types of reference into a single unified model. Heeman and Hirst model reference to objects that are copresent with the participants, or for which the participants have mutual knowledge. Our work has modelled reference to objects for which the participants have a large knowledge imbalance (the responder has no knowledge of the referent). There has also been research done on anaphoric reference, that is, reference to objects in a participant's focus of attention. Each of these types of reference has different underlying intentions, but all them can be, and are, used in discourse (including direction-giving dialogues). Heeman and Hirst's model is not a complete coverage of all the different ways a referring plan can clarified, and neither is ours for elaboration. By defining more discourse action schemas for judgement, refashioning, and suggestion, the coverage could be expanded and even generalized to handle even plans other than referring expressions.

A second avenue of research is to further investigate collaborative behaviour and the discourse structure that underlies it. We have modelled the acceptance process, a process essential and basic to collaboration, by using an attentional state that contains the contribution that is currently being considered. We have proposed a stack structure to contributions, which results in the most prevalent type of discourse structure. Unfortunately, the stack imposes a rigid order onto a discourse that is sometimes not seen in dialogues between humans (Clark and Schaefer (1989) investigate many structures underlying the acceptance of contributions). A more general theory of attention would say that the most important or salient contribution should be attended to, where importance would depend on the context of the discourse and the knowledge of the participants. This more general theory would subsume our model, where we equate importance with recency (i.e., the most recent contribution is considered first). A different metric for determining the importance of contributions needs to be developed to account for other less rigid discourse, where the acceptance processes for several contributions may happen in a different order than normal, be interleaved, or even be simultaneous.

A third direction would be to augment the model of knowledge and belief. The model we have adopted is minimal: belief revision is handled in a crude way, mutual belief is approximated by inference rules, and world knowledge is basic. Agents involved in a collaborative discourse have an interest in determining each other's beliefs so that they can cooperate and not impede the discourse. Establishing mutual belief and a common ground is fundamental to collaborating. A more extensive model of belief would be necessary before this model of collaborative behaviour could handle general task-oriented dialogue.

Bibliography

Allen, J. F. and Perrault, C. (1980). Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178. Reprinted in (Grosz, Sparck Jones and Webber, 1986).

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

Appelt, D. E. (1985b). *Planning English Sentences.* Cambridge University Press, Cambridge.

Appelt, D. E. (1985c). Some pragmatic issues in the planning of definite and indefinite noun phrases. In *Proceedings of the 23th Annual Meeting of the Association for Computational Linguistics*, pages 198–203.

Appelt, D. E. and Kronfeld, A. (1987). A computational model of referring. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 640–647.

Austin, J. L. (1962). *How To Do Things With Words.* Oxford University Press, Oxford.

Boden, D. and Zimmerman, D. H., editors (1991). *Talk and Social Structure.* Polity Press, Cambridge, England.

Carletta, J. (1990a). An architecture facilitating repair and replanning in interactive explanations. Technical Report No. 478, Dept. of Artificial Intelligence, Univ. of Edinburgh.

Carletta, J. (1990b). An incremental dialogue planner that monitors execution. Technical Report No. 480, Dept. of Artificial Intelligence, Univ. of Edinburgh.

Chu, J. (1993). Responding to user queries in a collaborative environment. In *Proceedings of the 31th Annual Meeting of the Association for Computational Linguistics*, pages 280–282.

Clark, H. H. and Marshall, C. (1981). Definite reference and mutual knowledge. In Joshi, A., Webber, B., and Sag, I., editors, *Elements of Discourse Understanding*, pages 10–63. Cambridge University Press, New York.

Clark, H. H. and Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science*, 13:259–294.

Clark, H. H. and Wilkes-Gibbs, D. (1986). Referring as a collaborative process. *Cognition*, 22:1–39. Reprinted in (Cohen, Morgan and Pollack, 1990, pp. 463–493).

Cohen, P. R. (1981). The need for referent identification as a planned action. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence (IJCAI-81)*, pages 31–36.

Cohen, P. R. (1984). Referring as requesting. In *Proceedings of the 10th International Conference on Computational Linguistics (COLING-84)*, pages 207–211.

Cohen, P. R. and Levesque, H. (1985). Speech acts and rationality. In *Proceedings of the 23th Annual Meeting of the Association for Computational Linguistics*, pages 49–59.

Cohen, P. R. and Levesque, H. J. (1990). Rational interaction as the basis for communication. In Cohen, P. R., Morgan, J., and Pollack, M., editors, *Intentions in Communication*, pages 221–255. MIT Press, Cambridge, Mass.

Cohen, P. R., Morgan, J., and Pollack, M., editors (1990). *Intentions in Communication*. MIT Press, Cambridge, Mass.

Cohen, P. R. and Perrault, C. (1979). Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212. Reprinted in (Grosz, Sparck Jones and Webber, 1986).

Cole, J., Frisch, A., Green, G., Hinrichs, E., Morgan, J., and research assistants (1991). Research on automated driving instructions: the second year. Technical report, Laboratory for Computational Linguistics, Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign, Urnana, IL.

Dale, R. (1987). Cooking up referring expressions. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 68–75.

Davis, J. R. (1989). *Back Seat Driver: Voice Assisted Automobile Navigation*. PhD thesis, Massachusetts Institute of Technology, Boston, Mass.

Devlin, A. S. (1976). The "small town" cognitive map: Adjusting to a new environment. In Moore, G. and Golledge, R., editors, *Environmental Knowing: Theories, Research and Methods*. Dowden, Hutchinson and Ross, Stroudsburg, Pa.

Downs, R. M. and Stea, D. (1973). Cognitive maps and spatial behaviour: Process and products. In Downs, R. and Stea, D., editors, *Image and Environment: Cognitive Mapping and spatial behaviour*, pages 8–26. Aldine Pub. Co., Chicago.

Frisch, A. M., Gerdenman, D., Griffith, J., Hinrichs, E. W., Morgan, J. L., Russell, D. W., and Sutanto, H. (1990). Automated natural language generation in a driving instruction system. In *Proceedings of the 1990 Artificial Intelligence Applications Symposium*, Phoenix, AZ.

Grosz, B. J. and Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

Grosz, B. J. and Sidner, C. L. (1990). Plans for discourse. In Cohen, P. R., Morgan, J., and Pollack, M., editors, *Intentions in Communication*, pages 417–444. MIT Press, Cambridge, Mass.

Grosz, B. J., Sparck Jones, K., and Webber, B. L., editors (1986). *Readings in Natural Language Processing*. Morgan Kaufmann Publishers.

Heeman, P. A. (1991). A computational model of collaboration on referring expressions. Technical Report CSRI-251, Computer Systems Research Institute (University of Toronto), Toronto, Canada. (MSc Thesis).

Heeman, P. A. (1993). Speech actions and mental states in task-oriented dialogues. In *Working Notes AAAI Spring Symposium on Reasoning about Mental States: Formal Theories & Applications*, pages 68–73, Stanford.

Heeman, P. A. and Hirst, G. (1992). Collaborating on referring expressions. Technical Report TR 435, Computer Science Dept., Univ. of Rochester, Rochester, New York.

Jefferson, G. (1972). Side sequences. In Sudnow, D., editor, *Studies in Social Interaction*, pages 294–338. Free Press, New York.

Kuipers, B. (1978). Modeling spatial knowledge. *Cognitive Science*, 2:129–153.

Lambert, L. and Carberry, S. (1991). A tripartite plan-based model for dialogue. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 47–54.

Litman, D. J. and Allen, J. F. (1987). A plan recognition model for subdialogues in conversations. *Cognitive Science*, 11(2):163–200.

Lochbaum, K. E., Grosz, B., and Sidner, C. (1990). Models of plans to support communication: an initial report. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-90)*, pages 485–490.

Lynch, K. (1960). *The Image of the City*. MIT Press, Cambridge, MA.

McDermott, D. (1980). A theory of metric spatial inference. In *Proceedings of the First Annual National Conference on Artificial Intelligence (AAAI-80)*, pages 246–248.

McDermott, D. and Davis, E. (1984). Planning routes through uncertain territory. *Artificial Intelligence*, 22(2):107–156.

McRoy, S. W. (1993). *Abductive Interpretation and Reinterpretation of Natural Language Utterances*. PhD thesis, University of Toronto.

Pattabhiraman, T. and Cercone, N. (1990). Selection: Salience, relevance and the coupling between domain-level tasks and text planning. In *Fifth International Workshop on Natural Language Generation*, pages 79–86, Dawson, PA.

Pollack, M. (1990). Plans as complex mental attitudes. In Cohen, P. R., Morgan, J., and Pollack, M., editors, *Intentions in Communication*, pages 77–103. MIT Press, Cambridge, Mass.

Psathas, G. (1986). The organization of directions in interaction. *Word*, 37:83–91.

Psathas, G. (1991). The structure of direction-giving in interaction. In Boden, D. and Zimmerman, D. H., editors, *Talk and Social Structure*, pages 195–216. Polity Press, Cambridge, England.

Psathas, G. and Kozloff, M. (1976). The structure of directions. *Semiotica*, 17(2):111–130.

Reiter, E. (1990). The computational complexity of avoiding conversational implicatures. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 97–104.

Reiter, E. and Dale, R. (1992). A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 232–238.

Riesbeck, C. K. (1980). "You can't miss it!": Judging clarity of directions. *Cognitive Science*, 4:285–303.

Searle, J. R. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge.

Svartvik, J. and Quirk, R., editors (1980). *A Corpus of English Conversation*. C.W.K. Gleerup, Lund.

Appendix A

Glossary

This glossary contains definitions for the various predicates that we use in our system. The predicates fall into three categories. First, *propositions* are similar to Prolog facts (or goals), except that an agent has beliefs about propositions. Second, *mental actions*, similar to Prolog goals, can only be used in the decomposition of the action schemas. Their satisfaction either instantiates the arguments or affects the mental state of the agent by changing or adding beliefs (side-effects). Third, a few predicates are considered common knowledge, and are similar to Prolog goals. These predicates are not mental actions, but are used as constraints and where-clauses.

Belief

`bel(Agt,Prop)`: `Agt1` believes that Prop is true.

`mb(Agt1,Agt2,Prop)`: `Agt1` and `Agt2` mutually believe that `Prop` is true. An agent can infer mutual belief if `bel(Agt1,Prop)` and `bel(Agt1,bel(Agt2,Prop))`.

`ab(Agt1,Agt2,Prop)`: `Agt1` and `Agt2` have an alternating belief that `Prop` is true. An agent can infer an alternating belief if `bel(Agt1,Prop)` or `bel(Agt1,ab(Agt2,Agt1, Prop)`.

Confidence

`confidence-headnoun(C,Entity,Category)`: A mental action that determines the confidence value `C` of referring to a discourse entity `Entity` of category `Category`. This action accesses an agent's category salience hierarchy.

`confidence-attrib(C,Entity,Pred)`: A mental action that determines the confidence value `C` of using an attribute `Pred` of the entity `Entity`. This action accesses an agent's property salience hierarchy.

`confidence-attrib-rel(C,Entity,RelEntity,Pred)`: A mental action that determines the confidence value `C` of using the attribute `Pred` that describes `Entity` relative to the other entity `RelEntity`. This action also accesses the agent's property salience hierarchy.

`confidence-add(C,C1,C2)`: A mental action that combines two confidence values, `C1` and `C2`, to form a new confidence value `C`. Simple arithmetic addition is used.

`confidence-exceed(Entity,C)`: A mental action that succeeds if the confidence value `C` exceeds the confidence threshold for a reference to the entity `Entity`.

`confidence-threshold(Cat,Threshold)`: An object of category `Cat` has the confidence threshold of `Threshold`.

Focus

`focus(Plan,SubPlan,Goal)`: The subplan `SubPlan` of `Plan` intended to achieve `Goal` is a focus element.

`focus-current(Focus)`: The focus element `Focus` encodes the subplan in focus.

Plan Derivation Predicates

`content(Plan,N,C)`: Node named by `N` of `Plan` has content `C`.

`constraint(Plan,A,C)`: Node `C` is a constraint of action node `A` of `Plan`.

`step(Plan,A,S)`: Node `S` is a step in the decomposition of action node `A` of `Plan`.

`yield(Plan,A,Y)`: The set of primitive actions `Y` is the yield of node `A` of `Plan`.

Plan Repair

`construct(Goal,Plan,SActs,Actions)`: A mental action that constructs a plan `Plan` that achieves goal `Goal` that contains the set of primitive actions `SActs` in its yield of `Actions`, if possible.

`substitute(Plan,Node,NewPart,NewPlan,NewActions)`: A mental action that undoes all variable bindings in `Plan` (except those in primitive actions that are not object terms of discourse entities), and then substitute content of `Node` in `Plan` by `NewPart`. The result of this is the plan `NewPlan` and the new primitive actions `NewActions`. (See Heeman and Hirst (1992).)

`evaluate(Plan)`: A mental action that succeeds if `Plan` is valid.

`replace(Plan,NewPlan)`: The plan derivation `NewPlan` replaces `Plan`.

Plans and Goals

`speaker(Agt)`: `Agt` is the current speaker. (A Prolog goal.)

`hearer(Agt)`: `Agt` is the current hearer. (A Prolog goal.)

`cstate(Agt1,Agt2,Plan,Goal)`: `Agt1` is engaged in a collaborative activity with `Agt2`. They are currently considering the domain plan `Plan` to achieve the goal `Goal`.

goal(Agt,Goal): Agt has goal Goal. Agents act to make their goals true.

plan(Agt,Plan,Goal): Agt has plan derivation Plan that is intended to achieve Goal. The agent may not believe that the plan is valid.

achieve(Plan,SubPlan,Goal): Executing subplan SubPlan of Plan causes Goal to be true.

entity(Id,Obj): The discourse entity used to represent the referent of the referring expression being built. Id is a unique identifier, and Obj is the referent object. The discourse entity is only used within other propositions and mental actions. (See Heeman and Hirst (1992).)

error(Plan,SubPlan,Node): Subplan SubPlan of Plan has an error at node Node. That is, SubPlan is invalid.

Reference

entity(Id,Obj): The discourse entity used to represent the referent of the referring expression being built. Id is a unique identifier, and Obj is the referent object. The discourse entity is only used within other propositions and mental actions. (See Heeman and Hirst (1992).)

ref(Entity,Obj): A mental action that unifies Obj to the object term of the discourse entity Entity. If the identifier term of Entity is not bound, this action will create a unique identifier for it and will make the value of Obj the referent.

identify(Agt1,Agt2,Entity): Agt1 is confident that she will be able to identify the referent that Agt2 associates with the discourse entity Entity.

set-context(Place1,Place2,Path,Orient): A mental action that sets the current reference context for the next reference by considering all places (including landmarks and intersecting paths) on Path between Place1 and Place2 given the orientation Orient. This action accesses the topological structure.

set-context(Place): A mental action that sets the current reference context by considering all objects, including paths that enter or leave, at Place. The two set-context actions are specific to the direction-giving domain.

context-current(Context): A mental action that determines the set of objects, Context, in the current reference context. A reference is made with respect to this set.

Route Description

knowroute(Agt1,Agt2,Route): Agt1 knows the route Route that Agt2 describes. Knowing a route is to know a sequence of actions that link the origin to the destination, and to be confident in the adequacy of all the references to paths and places made in the route description.

route-origin(Origin): Origin is the origin of the route being described. The origin is assumed to be common knowledge before a route description is made. (A Prolog goal.)

`route-dest(Dest)`: `Dest` is the destination of the route being described. The destination is also assumed to be common knowledge before a route description is made. (A Prolog goal.)

`what-route(Route)`: `Route` is the route that the direction giver will describe. (A Prolog goal.)

Salience

`salient-attrib(Entity,Pred,UsedPred,NewUsedPred)`: A mental action for determining a lambda expression `Pred` that encodes the most salient attribute for the category of the entity `Entity` that is not in the list `UsedPred`. The action appends `Pred` to `UsedPred` giving the list `NewUsedPred`.

`salient-attrib-rel(Entity,RelEntity,Pred,UsedPred,NewUsedPred)`: A mental action similar to `salient-attrib` except that the most salient attribute that describes `Entity` relative to the other entity `RelEntity` is chosen.

Set Operators

`subset(Set,Lambda,Subset)`: A mental action that computes the subset `Subset` of the set `Set` that satisfies the lambda expression `Lambda`. (A Prolog goal.)

`seteq(Set1,Set2)`: `Set1` is equal to `Set2` (i.e., they are subsets of one another). (A Prolog goal.)

`subset(Set1,Set2)`: `Set1` is a subset of `Set2`. (A Prolog goal.)

`psubset(Set1,Set2)`: `Set1` is a proper subset of `Set2`. (A Prolog goal.)

`partition(Set,Set1,Set2)`: An action that partitions `Set` by removing all elements of `Set1` from `Set` to give `Set2`. (A Prolog goal.)

Suggestion

`suggest(Plan,SubPlan,Actions)`: The set of actions `Actions` is a suggestion to expand subplan `SubPlan` of `Plan` by using the actions.

`salience-suggest-actions(Plan,SubPlan,Entity,Actions)`: A mental action that returns the set of primitive actions `Actions` that is the yield of a `modifier` plan for the most salient modifier not used in the subplan `SubPlan` of `Plan`. This action accesses an agent's property salience hierarchy.

World Knowledge

`category(Object,Cat)`: The object with internal name `Object` is of category `Cat`.

`called(Object,Name)`: The object with internal name `Object` has the external name `Name`.

`has(Object,Landmark)`: The object with internal name `Object` has the landmark (also an object) with internal name `Landmark`.

`place(Object,Intersection,LocalGeometry)`: The place with internal name `Object` is the intersection of the paths whose names are listed in `Intersection`. `LocalGeometry` specifies the angle and orientation of each path that enters of leaves the intersection.

`path(Object,Row)`: The path with internal name `Object` has the places listed in `Row` on it in that order.