February 1992

Goals and Actions in Natural Language Instructions

Barbara Di Eugenio
University of Pennsylvania

Goals and Actions in Natural Language Instructions

Abstract

Human agents are extremely flexible in dealing with Natural Language instructions: they are able both to adapt the plan they are developing to the input instructions, and vice versa, to adapt the input instructions to the plan they are developing. Borrowing the term from [Lewis 1979], I call this two-way adaptation process *accommodation*. In this proposal, I first define accommodation in the context of processing instructions. I then provide evidence for the particular inferences I advocate, and for the further claim that such inferences are directed by the *goal* to achieve which certain action is performed. The evidence I provide comes from my analysis of naturally occurring instructions, and in particular of *purpose clauses* and of *negative imperatives*. Finally, I propose a computational model of instructions able to support accommodation inferences. Such model is composed of: a speaker / hearer model of imperatives, based on the one presented in [Cohen and Levesque 90]; an action representation formalism based on a hybrid system, á la KRYPTON [Brachman *et al.* 1983a], whose primitives are those proposed in [Jackendoff 1990]; and inference mechanisms that contribute to building the structure of the intentions that the agent develops while interpreting instructions.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No.MS-CIS-92-07.

Goals and Actions In Natural Language Instructions

MS-CIS-92-07
LINC LAB 213

Barbara Di Eugenio

University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

February 1992

GOALS AND ACTIONS IN NATURAL LANGUAGE INSTRUCTIONS[*]

Dissertation Proposal

Barbara Di Eugenio

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA
dieugeni@linc.cis.upenn.edu

Abstract

Human agents are extremely flexible in dealing with Natural Language instructions: they are able both to adapt the plan they are developing to the input instructions, and vice versa, to adapt the input instructions to the plan they are developing. Borrowing the term from [Lewis, 1979], I call this two-way adaptation process *accommodation.*

In this proposal, I first define accommodation in the context of processing instructions. I then provide evidence for the particular inferences I advocate, and for the further claim that such inferences are directed by the *goal* to achieve which a certain action is performed. The evidence I provide comes from my analysis of naturally occurring instructions, and in particular of *purpose clauses* and of *negative imperatives.*

Finally, I propose a computational model of instructions able to support accommodation inferences. Such model is composed of: a speaker / hearer model of imperatives, based on the one presented in [Cohen and Levesque, 1990b]; an action representation formalism based on a hybrid system, á la KRYPTON [Brachman *et al.*, 1983a], whose primitives are those proposed in [Jackendoff, 1990]; and inference mechanisms that contribute to building the structure of the intentions that the agent develops while interpreting instructions.

[*]This research is supported by DARPA grant no. N0014-90-J-1863 and ARO grant no. DAALO3-89-C0031PR1.

Contents

List of Figures

Chapter 1

Introduction

The problem of understanding and executing Natural Language instructions or commands is of both theoretic and practical interest: theoretic, because it bears on cognitive theories of actions; practical, because almost since the beginning of Computer Science there has been a desire to communicate with the computer in Natural Language, and have it execute commands.

Instructions are meant to affect behavior, namely, the actions the agent performs; many researchers, among others Winograd [1972], Chapman [1991], Cohen and Levesque [1990b], Vere and Bickmore [1990], Alterman et al. [1991], have been and are working on the problem of mapping Natural Language instructions onto an agent's behavior. Many complex facets of this problem have been addressed, for example identifying the intentions that the agent should adopt as a result of a certain instruction, or computing temporal constraints among different actions the agent has to perform. However, an aspect that, strangely enough, no one has really addressed is *computing the objects* of the intentions the agent adopts, namely, the actions to be performed: in general, researchers have simply equated such objects with predicate-argument structures extracted from the parsed input string. This may perhaps be correct when dealing with simple positive imperatives, which constitute almost the whole of the instructions examined in the literature. However, naturally occurring instructions are complex, and action descriptions cannot just be *extracted* from them, but they need to be computed. As a case in point, consider:

Ex. 1 *a) Place a plank between two ladders.*
b) Place a plank between two ladders to create a simple scaffold.

In both a) and b), the action to be executed is *"place a plank between two ladders"*. However, 1.a would be correctly interpreted by placing the plank *anywhere* between the two ladders: this shows that in b) the agent must be inferring the proper position for the plank from the expressed goal *"to create a simple scaffold"*. Notice therefore that the goal an action is meant to achieve constrains the interpretation and / or execution of the action itself. Furthermore, notice how flexible a human agent is, in that s/he can deal with action descriptions at different levels of specificity without any difficulty: s/he is able to adapt her knowledge to the input, and vice versa, to interpret the input instructions according to her knowledge.

I hope the previous example gives the reader a flavor for the three main points that I want to make in this proposal:

1. The actions an agent has to perform when s/he is given instructions have to be *computed*

from the descriptions given in the instructions themselves, as opposed to simply *extracted* from such descriptions.

2. The goals that an agent is trying to achieve guide this computation; many of these goals are explicitly stated in the instructions themselves.

3. Action descriptions found in instructions can't be expected to exactly match the knowledge that an agent has about actions and their characteristics. Therefore, to model an agent interpreting instructions we need a flexible action representation, inference mechanisms that can deal with actions descriptions at various levels of specificity, and that build a structure of the agent's intentions that will help interpret the subsequent instructions. Borrowing the term from [Lewis, 1979], I collectively call these inference processes *accommodation.*

1.1 Thesis Statement

More in detail, my claim is that

most instructions don't exactly mirror the agent's knowledge, but are understood by accommodating them in the context of the general plan the agent is considering; the agent's accommodation process is guided by the goal(s) that s/he is trying to achieve. The concept of goal itself is pervasive in NL instructions, and a NL system which interprets instructions must be able to recognize and/or hypothesize goals, keep track of them, and use them in computing the description of the action to be performed.

- To show that my claim is *justified*:
    1. I will illustrate my views on the agent's inference processes by means of the analysis of naturally occurring data [1] – in particular, the two constructions of *purpose clauses* and *negative imperatives.* I will show what pragmatic functions they perform, and what kind of inferences an agent has to do to understand them.
    2. I will show how the agent's inferences are directed by the notion of *goal.* This is equally true when the agent interprets *purpose clauses*, that express goals explicitly; or *negative imperatives*, whose correct interpretation is often constrained by the goal in the context of which an action should *not* be performed.
- To show that my claim is *valid*, I will discuss the consequences of such views for a computational model of instructions. I will propose to develop a model consisting of
    1. A speaker / hearer model of imperatives, based on the one presented in [Cohen and Levesque, 1990b].
    2. A representation of actions, and of goals, that provides support to perform the inferences I will discuss. I will propose that a hybrid system, á la KRYPTON [Brachman *et al.*, 1983a] should be used, whose primitives are those that Ray Jackendoff proposes in [1990] for the semantic representation of verbs and actions.

[1] All the examples I will discuss are naturally occurring ones, unless otherwise noted.

3. Inference mechanisms that contribute to building the structure of the intentions – the *plan graph* – that the agent develops while interpreting instructions; the representation of such plan graph will also be discussed.

1.2 Outline of Proposal

In ch. 2, I develop my views on the process of understanding instructions: after reviewing some of the approaches taken in the literature, I will define what I mean by *accommodating instructions*, and how accommodation relates to *plan recognition*. At the end of that chapter, I will restate my main claim, the way I will assess it, and the work I propose to do.

Chapters 3 and 4 are devoted to the analysis of the data, *purpose clauses* and *negative imperatives*, that provide evidence for the view of instructions put forward in ch. 2.
In ch. 3, I examine purpose clauses, showing how *goals*, that purpose clauses explicitly describe, constrain the interpretation of action descriptions; and I describe which relations between the two actions described respectively in the matrix and in the purpose clause are possible.
In ch. 4, I analyze negative imperatives. I show that there exist two classes of negative imperatives, that have different pragmatic functions. Negative imperatives provide further support to the claims that goals affect the interpretation of action descriptions, and that action descriptions need to be computed.
I conclude both chapters with some remarks on the consequences that the analysis of such data has for a computational model of instructions.

In ch. 5, I will describe the steps I have already undertaken, and the work I propose to do to define a computational model of instructions: I will present the speaker / hearer model, the action representation formalism, the inference processes that build the plan graph, and the plan graph itself. In this chapter I will also describe the application context in which my work is taking place, namely, the *Animation from Natural Language* project at the University of Pennsylvania.

Ch. 6 is devoted to summary and conclusions.

Chapter 2

On instruction understanding

The problem of understanding and executing Natural Language instructions can be looked at from many different points of view: my particular interest lies in the inferences that an agent performs upon being given instructions in Natural Language – NL for short. Actually, a more correct characterization of such inferences would be "those that an agent *may* perform upon being given NL instructions". I am using *may* because it is not clear how interpretation and execution of instructions can be totally teased apart: namely, when are these inferences performed, when the agent interprets the instruction, or only when he executes it? The point I want to stress is that there are inferences that an agent can perform without being yet *engaged* in carrying out the instructions: these are the ones I would like to account for. I will elaborate on this point in sec. 2.1.2.

Discussing inferences that the agent performs upon getting NL instructions clearly cannot be done independently from one's general approach to instruction understanding. Therefore, in the following section, I will show what understanding instructions has been taken to be by different researchers, then I will start developing my own theory of understanding instructions, based on Lewis's notion of accommodation [Lewis, 1979].

2.1 Accommodating instructions

There are many different points of view from which the problem of interpreting and executing instructions may be examined: they depend on different factors, among them whether the main research goal is to develop a cognitive theory of how humans interpret and execute instructions, or to implement a working program; and the type of scenario in which instructions have to be interpreted and executed. Clearly instructions are meant to affect behavior, namely, the actions the agent performs; in turn, the choice of actions to perform depends on the goals the agent adopts, and on the performance situation he finds himself in. Unfortunately, in the literature one finds some confusion about how instructions relate to an agent's *goals* and *actions*. In the following, I will try to characterize existing approaches to instruction understanding on the basis of

1. whether an instruction [1] is taken to be a description of an action to be performed, or a pointer to one of the agent's internal routines: in the former case, the agent has to actively compute the object of the corresponding intention, in the latter, s/he has to execute the corresponding routine;
2. how the instruction relates to the goals that the agent may already have, and vice versa, how such preexisting goals can affect the interpretation of the instruction itself;
3. whether the instructor has total control over the agent's actions and / or goals;
4. whether the communicative situation in which an instruction is issued and the performance situation in which it is executed coincide.

Before describing the different approaches that have been taken so far, I would like to elaborate on the first two items mentioned above, whether an agent maps instructions onto goals to perform an action, and whether there is any relation between an instruction and the goals an agent may already have.

First of all, one should be clear about what *action* and *goal* are taken to mean. One could say that the goal is a state to be achieved and the action the act whose performance results in such a state; or that the goal requires intentionality on the part of the agent, while the action doesn't. For example Cohen and Levesque in [1990b] define goals as *chosen desires*; an agent has a *persistent goal* if he has a goal that he currently believes to be false and that he will continue to choose. Given that instructions describe the world in terms of actions, much more than in terms of desirable states, I think a reasonable view to take, and in fact one that has been widely adopted, is that an agent maps an instruction onto a goal, the goal of *performing the action described in the instruction.* However, this point of view is at best adequate for situations characterized by an obedient agent and by instructions limited to simple positive imperatives. Consider a command like *don't eat the cookies.* First of all, the agent may adopt three different attitudes toward such a goal:

No commitment. If such an instruction is given to a passive agent, or to an agent who is not interested in eating cookies at all, because e.g. he does not eat anything with sugar at any rate, the correct final state will be achieved even if the agent's chosen desires, to use Cohen and Levesque's terminology, don't include *not eating the cookies.*

Rejection. The agent can of course reject the proposed goal, and in fact do the opposite.

Commitment. The agent does adopt the *persistent goal* of not eating the cookies.

Let's suppose now the agent does adopt that goal, either because we deal with an obedient slave, as many computer systems are built to be, or because, more realistically, the agent is cooperative – see [Cohen and Levesque, 1990b] for a characterization of a cooperative agent. Adopting the goal of not eating the cookies can result in (at least) three different kinds of behavior, none of which in fact can be described as *performing the action described in the*

[1] I will actually limit myself to imperatives: one does find declarative statements in instructions, especially in instruction manuals, but I won't deal with them. Mostly, declarative statements appear in the introductory part of the manual, where tools and materials are described, not in the part describing the actual steps of the task at hand.

instruction – in fact, one may wonder whether negative instructions describe actions to be performed at all:

No observable action. The agent doesn't do anything that changes the state of the world with respect to the cookies.

Constraints on other actions. Such a persistent goal may result in constraints on other goals that the agent may adopt: for example, if the agent were told *Go and get me a glass of water*, he could answer *I'd rather not, because I would have to go to the kitchen, the cookies are there, and the temptation would be too strong to resist.*

Related observable action. The agent may think that he cannot yield to the temptation of eating the cookies, and therefore he may lock them away, and put the key in a place which is difficult to reach: in this case the agent does perform some actions that help him achieve the desired goal.

The point I want to make is that many researchers have concentrated their efforts on positive commands, which one can indeed consider as *describing an action α to be performed*: a cooperative agent will then adopt the goal Γ = PERFORM (H, α)). However, as we will see, Γ varies according to the type of imperative an agent is presented with – a complex positive imperative including a purpose clause, or a negative one, or a warning. Even when Γ is indeed *performance of the described action*, as in positive commands, the *described action α* cannot be taken from the instruction as it is and just "plugged in" to obtain something like GOAL(H, PERFORM (H,α)): α has to be computed, in ways that I will discuss in this proposal.

That Γ is not simply PERFORM (H, α) and that, at any rate, α has to be computed are two of the main points I want to address. Before talking about them, though, I will discuss five different approaches to instruction understanding in light of the previous comments.

"Simon says" approaches. In the *Simon says* game, each command issued by the leader ("Simon says put your hands on your ears") is meant to lead directly and immediately to behavior on the part of the player. In this case, a command is mapped onto a goal to perform the action it describes, and no further reasoning on the part of the agent is necessary, because the described actions are *basic*, namely, physical actions [2]. There are no other goals that the agent has, the instructor has total control over the goals that the agent adopts, and the communicative situation is the performance situation.

In computer systems of this kind, a command corresponds to a stored program: once the command has been interpreted, the corresponding program is executed. An example of such systems is the by now venerable SHRLDU [Winograd, 1972]. The "Simon says" approach should not be discarded as the "easy" approach to instruction understanding, because the interpretation and execution processes may be quite complicated: for example in SHRLDU, actions other than those described in the input command may be performed if they are necessary to achieve preconditions of the described ones.

The situated approach. Chapman in [1991] proposes a model of instruction interpretation that is concerned with reacting to the situation at hand, and that heavily relies on perception. The instructor gives instructions or offers suggestions, while watching an agent

[2]See [Goldman, 1970] and [Pollack, 1986] for a discussion of *basic* actions.

engaged in an activity – in Chapman's thesis, the activity is playing a video-game. There is no distinction between the agent's actions and goals, and each action can be adopted as a goal and executed. However, the agent is not simply an obedient slave and can choose whether to execute a certain action. Choices are hard-wired in an arbitration network.

Chapman's view of instructions is that they give advice, but also that they are not much more than perceptual stimuli.

> When Sonja [the agent playing the game] is given an instruction, it registers the entities the instruction refers to and uses the instruction to choose between courses of action that themselves make sense in the current situation. An instruction can fail to make sense if it refers to entities that are not present in the situation in which it is given, or if the activity it recommends is implausible in its own right. ... As part of taking an instruction you register aspects of the situation you weren't previously aware of, and project new possibilities. Based on these new registrations you engage in new activities. ... Instructions recommend courses of action; Sonja's arbitration network takes account of such recommendations. ... Because instructions can only influence arbitration when they make sense in terms of activities Sonja could engage in autonomously, their role in Sonja is one of *management* only. [Chapman, 1991, p.76].

An instruction then identifies one action among the ones that belong to the system repertoire, and gives an additional reason why the agent should choose it. However, that action will not necessarily be adopted as a goal. The communicative situation and the performance situation coincide as in the "Simon Says" approach, modulo *making sense* of the situation, e.g. *turn left* does not mean *turn left immediately*, but *turn left in the first place where you can do so.* Chapman's approach seems more concerned with making (immediate) decisions about behavior, than with deriving knowledge from instructions and then acting.

Vere and Bickmore. Vere and Bickmore in [1990] describe the architecture of *a complete integrated agent.* Their agent, a little submarine named Homer, is able to navigate in a harbor full of other objects, to go to places, to deliver objects from one place to another. The submarine can take part in dialogues that include questions, assertions and commands. *A command is a special case of an INFORM event in which the information transmitted is that the informer has a goal and wants the informee to achieve it for him. ... There is a demon in the reflective processes which reacts to general commands by extracting the goals, processing them, and sending them to the temporal planner for plan synthesis.*

Vere and Bickmore's view is the usual one, that a command is mapped into a goal to perform the described action – apart from negative commands, see below. Homer accepts simple commands from the instructor, who has total control over it, and adopts them as its own goals: then it reasons about whether it can achieve them. One interesting point is that there can be time constraints attached to commands, either explicitly – *Drop the package at the barge next Sat. at 9 p.m.*, or implicitly – *If you see an animal tomorrow, photograph it*: therefore the utterance situation and the performance situation don't coincide.

The system allows for *replanning*, which is used to deal with new declarative information which changes the parameters under which a plan had been formulated, and with additional goals, that may have to be achieved before another goal for which planning had already taken place. Homer does accept negative commands, which are considered as global constraints on the activities it can perform. For example, if Homer is told *Don't leave the island today*, and later, the same day, *Take a picture of the Codfish*, it will say it cannot comply with the latter instruction, because to do so it would be necessary to leave the island.

Although Homer's abilities are quite impressive, it is not clear whether replanning and the capacity of dealing with negative commands amount to something more than propagating temporal constraints.

The cooperative agent. Cohen and Levesque in [1990b] adopt the view that the agent adopts a certain goal as the result of an imperative because he is cooperative.

> If an imperative is uttered in circumstances where a hearer does not suspect (at some level of alternating belief) that the speaker is insincere in his desire to get an addressed person to believe that he has a certain goal, then the hearer will believe (at that level) that the speaker really has that goal. The goal in question here is that the addressed person should perform something that fulfills the condition expressed by the imperative sentence (p. 237). ... If addr [the addressee] does not mind doing the action and is helpfully disposed toward spkr [the speaker], then we can conclude that addr intends to do the action relative to spkr's desire (p.238).

As it is apparent from the quotes above, Cohen and Levesque equate *perform something that fulfills the condition expressed by the imperative sentence* with *doing the action relative to spkr's desire*, which is actually formalized as a, where a is the content of the imperative. In light of my observations above on negative imperatives, this is not sufficient. Even limiting oneself to positive imperatives one may have to compute the described action a.

In a more recent paper [1991], Cohen and Levesque go into some details on how an agent comes to intend to do an action to achieve a goal, and they hint at an action representation language, built *inductively out of primitive events, and complex expressions created by action-forming operators for sequential, repetitive, concurrent, disjunctive, and contextual actions:* this seems to me evidence for the fact that they consider event descriptions to be primitive, namely, that they are not interested in the internal structure of such "primitive" events; besides, they still don't address at all the issue of computing which actions an instruction describes.

Cohen and Levesque don't discuss whether the utterance situation and the performance situation coincide: I imagine they would pay attention to such distinction if they did examine the performance of the action *that fulfills the condition expressed by the imperative sentence.*

In my opinion, the most relevant part of their contribution is the fact that they model the communicative act taking place between the speaker and the hearer: I will come back to this and related issues in ch. 5.

Instructions as a way of solving impasses. A different view on instructions is advocated in [Alterman *et al.*, 1991]. The stress is on extending stored knowledge to new but similar situations, as for example when knowledge about using a certain device – a home phone – is applied to the usage of a new, but similar device – a pay phone.

An agent has a basic set of skills and plans to solve certain problems. Instructions are used as a resource only when the stored knowledge about plans cannot be adapted to the situation at hand, namely, when the agent is blocked in mapping the goal to action: instructions come in to specify the appropriate action(s) that will satisfy the goal in that particular situation. The agent has to integrate these actions in his knowledge and to understand how they relate to the goal he was trying to achieve: this is the only paper I know which assumes a rich relation between the instructions and the preexisting goal(s). It is crucial that the communicative situation and the performance situation are the same: communication comes in to repair a breakdown in performance. From a certain point of view, Chapman's approach is not so different, in the sense that for Chapman as well instructions come in as an additional source of information about the course of action to follow, when the agent doesn't have enough information to choose a course of action over the other.

The following table summarizes the approaches outlined above: the column *goal contains action* refers to whether an instruction is **directly** mapped into the goal of performing the described action; NA stands for Not Applicable:

Approach	goal contains action	relation between instructions and other goals	instructor in control	utterance sit. necessarily = performance sit.
"Simon says"	Y	None	Y	Y
Chapman	NA	NA	N	Y
Vere, Bickmore	Y[a]	Limited	Y	N
Cohen, Levesque	Y	Not incompatible	N	N
Alterman	Y	Rich	N	Y

[a]Apart from negative commands.

I think one conclusion that can be drawn from this table is that in general researchers have not worried about computing the action α to be performed; and that, apart from [Alterman *et al.*, 1991], not much attention is paid to the mutual interaction between the current instruction and preexisting goals. The approach I am going to propose instead relies in an active way on such mutual interaction, and tries to make the agent's knowledge flexible enough so that the same instruction uttered in the context of different goals will result in different behavior.

The scenario I have in mind is as follows: the speaker – to whom I will refer by means of "S", and feminine pronouns – gives instructions to the hearer – "H", referred to by means of masculine pronouns. S formulates her instructions by taking into account what S thinks H knows and can do. Of course, S doesn't exactly know what H knows, and moreover, in certain applications, S issues commands related to S's, not H's, knowledge of the current situation. However, S can rely on some basic skills and abilities that S knows H has, and on the fact that H is not a cognitively passive slave that can accept only instructions that exactly mirror his knowledge, but can accommodate instructions with respect to his general knowledge, and his knowledge of the situation.

That H can take, and in fact has to take, an active part in the interpretation and execution of instructions has been shown in work done in various areas, among which anthropology and plan recognition.

As an example of observations made in an anthropological setting, consider [Suchman, 1987]; she says that instructions are *irremediably incomplete*, and that *the problem of the instruction-follower is viewed as one of turning essentially partial descriptions of objects and actions into concrete practical activities with predictable outcome. ... instructions rely upon the recipient's ability to do the implicit work of anchoring descriptions to concrete objects and actions.*

Some plan recognition work, in particular [Pollack, 1986], is also concerned with the fact that H has to be active in the execution of instructions. One of her examples is

> S: "I want to talk to Kathy. Do you know the phone number at the hospital? [3]"
> H: "She's already been discharged. Her home number is xxxxxx"

This dialogue fragment shows flexibility on H's part: he addresses S's goal of talking to Kathy, not so much her direct question, and answers cooperatively.

The kind of flexibility that I want to account for is more limited, in a sense, than what Martha Pollack wants to account for: I'll discuss the differences with Pollack's approach in sect. 2.1.3. My view is that the hearer's task is one of understanding the role that an input instruction plays in a global plan to do something, but also to *adapt* his own knowledge of that plan to the instruction he has to understand. I would contend that, in the vast majority of cases, instructions assume a certain basic set of skills and abilities of the agent and just specify how they are to be related to one another. I think this is consistent with what Chapman says, that *routine activity is by far the more common. We spend most of our lives engaged in activities like making breakfast, driving to work, reading the paper, ...; these rarely involve novel elements. Even creative work is mainly routine: it's rewording sentences or painting the background of the scene ....* While he was mainly concerned with how much you can do with just these basic skills, I am more concerned with how you can understand instructions by assuming these basic building blocks and relating them in the way the instructions suggest.

To illustrate what I mean with *relating basic blocks to each other*, let's consider the following example [4]:

Ex. 2 *Go into the other room to get the urn of coffee.*

It is clear that H doesn't have a particular plan that deals with *getting an urn of coffee.* He will have a generic plan about *get x*, which he will adapt to the instructions S gives him [5]. Some of this adaptation task is quite easy, namely, parameter instantiation, in this case binding the patient parameter of *get* to an individual satisfying the description *urn of coffee.* But the rest is not so direct: H has to find the connection between *go into the other room* and *get the urn of*

[3]That S formulates his request as a question is not important: *give me the phone number at the hospital* would be equivalent, albeit less polite.

[4]From the animation script used in the AnimNL project [Webber *et al.*, 1991] – see ch. 5.

[5]Actually H may have more than one single plan for *get x*, in which case *go into the other room* may in fact help to select the plan the instructor has in mind – I will address this issue later.

coffee. This connection requires reasoning about the effects of *go* with respect to the plan *get x*; notice that the (most direct) connection between these two actions requires the assumption that the referent of *the urn of coffee* is in *the other room.*

A possible strategy is as follows. The agent's plan for *get* will contain a requirement that the agent move to the place where the object to be "gotten" is located: from this the agent can derive that the (most direct) connection between these two actions is that *go into the other room* fulfills this requirement, under the assumption that the referent of *the urn of coffee* is in *the other room.*

This process of adaptation of H's plan to S's instructions, I would claim, is what normally happens when people interpret instructions. I will call this process *accommodation*, borrowing the term from [Lewis, 1979], and using it in a related but slightly different way, as I will show in the next two sections.

2.1.1 Lewis on accommodation

Lewis in [1979] uses the term *accommodation* to refer to the process by which *conversational score does tend to evolve in such a way as is required in order to make whatever occurs count as correct play.* By conversational score, Lewis means the state of the conversation, described by various components, such as sets of presuppositions.

Lewis uses the concept of accommodation to deal with different linguistic phenomena, among which

Presuppositions. If I say *All of Frederick's children are asleep*, and there has been no mention of Fred's children so far, the presupposition that Fred has children is readily added to the set of current presuppositions.

Definite descriptions. I will actually describe the way that Heim in [1982] exploits *accommodation* to deal with *novel definites*, because her treatment is more perspicuous.

In the file-change semantics Heim develops for NPs, the usage of a definite NP is felicitous only if there is already an appropriate card in the file describing the referent of that NP. However, there are some usages of definite NPs that do not fit with this condition, for example the inferrable usage ("the author" in "John read a book about Schubert, and wants to write to the author"). Heim rewrites her Novelty-Familiarity-Condition by postulating accommodation as *an adjustment of the file that is triggered by a violation of a felicity condition and consists of adding to the file enough information to remedy the infelicity.*

She also adds that *Under which conditions is accommodation an available option, and what exactly is added to the file when the option is taken? these questions are by no means easy to answer ...*

Planning. Lewis's accommodation is usually mentioned in regard to referring expressions, but he also talks about planning, albeit briefly. He says (notice that the utterance that has to be accommodated is uttered while discussing a plan to steal plutonium from a nuclear plant):

> Much as some things said in ordinary conversation require suitable presuppositions, so some things that we say in the course of planning require, for their acceptability, that the plan contains some suitable provisions. If I say "Then you drive the getaway car up to the side gate", that is acceptable only if the plan includes provision for a getaway car. That might or might not have been part of the plan already. If not, it may become part of the plan just because it is required by what I said.

As far as planning is concerned, therefore, Lewis's accommodation refers to a global process, very similar to the one occurring for presuppositions, of making room for (relatively) new things in the plan if they are not already there. I am talking about a similar form of accommodation, that allows an agent to understand instructions that don't exactly mirror his knowledge. As I will show in the next section, I think the fundamental concept that comes to play during the accommodation of α is the *purpose* to achieve which α has to be performed.

Before I proceed in the discussion, I need to draw some distinctions.

1. So far, I have been talking about accommodating *instructions*: clearly, what is accommodated is the content of an instruction. For example, if we map a positive command onto something like GOAL (H, PERFORM (α)), then what needs to be accommodated is actually the internal argument α.

2. In the next section, I will start talking about the *purpose* of an action, and for lack of a better terminology, I'll use the term *goal* in that case too. The notion of *goal* will then correspond to a three-place predicate, stating that the agent has the goal of performing α in order to perform β:

   $$\text{GOAL(H, PERFORM}(\alpha)\text{, PERFORM}(\beta)\text{)}$$ [6]

   For the moment, I'll use the term *goal* to refer to both definitions of the predicate GOAL: the reader should keep the distinction in mind.

2.1.2 The notion of goal as guide to accommodation

Two questions that Lewis does not address are: why does H accept a certain instruction, and accommodate it in his plan to do something? Does accommodation happen because there is something else that supports and justifies it?

My claim is that the answer to the latter question is yes, and that the basic notion here is that of *goal*. In Lewis's example above, the accommodation necessary to account for the *getaway car* is readily performed because there is a global goal for the plan in question, which is *steal plutonium from a nuclear plant*; and a subgoal of this plan is *escape safely*. It's not blind accommodation, then: it happens because there is something that justifies it. Notice that mentioning the function of the car, *getaway*, is crucial to the accommodation process: if the instruction to be accommodated had been *Then you drive the car up to the side gate*, it would have been much harder to find its place in the global plan, if no car had been mentioned so far.

[6]This definition should not be taken too literally: I am just trying to keep the two different notions of goal distinct. I will come back to formalizing them in later chapters.

In general, instructions cannot be interpreted without taking into account the context in which they are issued. This is of course true of other NL phenomena as well, such as referential expressions. However, in the case of instructions particular kinds of context have to be taken into account, namely, the speaker's beliefs, as Martha Pollack has shown [Pollack, 1986]; and the one I am most interested in, the *goal* in the context of which that particular instruction has to be interpreted and executed.

To prove my point, consider a simple instruction such as:

Ex. 3 *Open the door.*

The execution of this action will vary according to whether the goal of opening the door is letting the cat in or out – a small opening is sufficient; letting a guest in or out – the door has to be open wider than for the cat; letting in or out movers that carry bulky furniture – the door is best propped open.

In general, there is not a single goal according to which an instruction has to be interpreted and executed, but a set of them. Consider this instruction step in making a stuffed toy:

Ex. 4 *To attach arms, use doubled white carpet thread and push needle in one side of the body and out the other side at points indicated by dots on the pattern.*

The action *push needle in one side of the body ...* has to be performed *to attach arms.* In turn, the action *attach arms* has to be performed in the context of making a doll: this puts some requirements on *attaching arms*, e.g. where the arms have to be attached. These requirements "filter down" to the interpretation and execution of *push needle in one side of the body etc.* Notice that the modifier *at points indicated by dots in the pattern* is not sufficient to establish where to attach the arms. In fact, the pattern contains many dots, corresponding to where arms, legs, head should be attached: therefore, the goal *to attach arms* does affect the interpretation of the prepositional phrases *in one side of the body and out the other side.*

I would now like to conclude this section by fending off some possible criticisms.

1. **The function of goals is only that of giving the agent the reason why he should perform a certain action.** This is a common view, but naturally occurring purpose clauses show that there is something else going on. I already discussed the following example in ch. 1:

   Ex. 5 *a) Place a plank between two ladders.*
   b) Place a plank between two ladders to create a simple scaffold.

   In both a) and b), the action to be executed is *"place a plank between two ladders".* However, Ex. 5.b would be correctly interpreted by placing the plank *anywhere* between the two ladders: this shows that in a) the agent must be inferring the proper position for the plank from the expressed *goal "to create a simple scaffold".* Therefore, *to create a simple scaffold* isn't simply used to tell H why he should *place a plank between two ladders,* but also to help him execute the action correctly.

2. **Goals are significant only at execution time.** Someone may object that, while the concept of goal is definitely relevant, it has to be taken into account only at the moment

of executing an action. This is the view embodied in [Alterman *et al.*, 1991]. While this is true for Ex. 3 above, there are two reasons why one would think that goals affect the *interpretation* of instructions too. First of all, it is very often NL itself that conveys, more or less implicitly, the goal in the context of which an instruction should be interpreted, as in Exs. 5.a and 6. Therefore a NL system that interprets actions has to be able to recognize and actively use goals as they arise from NL specifications of actions, to compute the descriptions of the actions to be executed.

Secondly, an agent may perform inferences that make the actions he has to execute executable without being already engaged in the corresponding activity. An example will make my claim clearer:

Ex. 6 *Cut the square in half to create two triangles.*

This example contains the explicit goal *to create two triangles.* The action described in the matrix sentence, *cut the square in half*, is underspecified in that there are an infinite number of ways of cutting a square in half, including along an axis and along a diagonal. An agent with basic knowledge about the world, though, won't have any difficulty in executing this instruction in the correct way, namely, cutting the square along the diagonal, because the goal *to create two triangles* unambiguously specifies how the cutting action should be performed. What I want to point out here is that I am not claiming that the *computation* of the action to be executed necessarily happens at interpretation time; what I do want to stress though is that, given his basic abilities, the agent can understand what is required simply from the NL instruction. These are the kind of inferences I'd like to account for; and this is the sense in which I am saying that the *goal* affects not just execution, but also interpretation of instructions.

2.1.3 Differences with other approaches

I would now like to answer two very likely questions about my work: how is accommodation different from plan recognition [7]? and how is it different from the approach to instructions advocated in [Alterman *et al.*, 1991]?

1. **Is accommodation different from plan recognition?**
   Accommodation can be seen as a subtype of *plan recognition*, but only if *plan recognition* is defined is terms as general as those used by Kautz in [1990]: *One is given a fragmented, impoverished description of the actions performed by one or more agents, and expected to infer a rich, highly interrelated description. The new description fills out details of the setting, and relates the actions of the agents in the scenario to their goals and future actions. The result of the plan recognition process can be used to generate summaries of the situation, to help (or hinder) the agent(s), and to build up a context for use in disambiguating further observations.*

   Accommodation, as plan recognition, can be considered as a process of *inferring a rich, highly interrelated description*, even if accommodation deals with descriptions of future

[7] I am here talking about accommodation processes relative to understanding instructions, not about the accommodation in general, e.g. with respect to accommodating presuppositions.

behavior, plan recognition with descriptions of observed events. However, the term *plan recognition* is generally used to refer to the set of inferences that various researchers working in the area have investigated. Accommodation is indeed different from those particular implementations of plan recognition: to explain the difference, I will first illustrate the inferences I think are part of accommodation, and then the inferences investigated by two of the most important researchers in the area of plan recognition, Kautz and Pollack.

Accommodation. The inferences that I have identified so far as being part of accommodation can be typified by Exs. 6 – *Cut the square in half to create two triangles* and 2 – *Go into the other room to get the urn of coffee.*

In the first kind of accommodation – Ex. 6 – we have to find a more specific action α_1 which will achieve the goal specified by the purpose clause, as shown in Fig. 2.1.

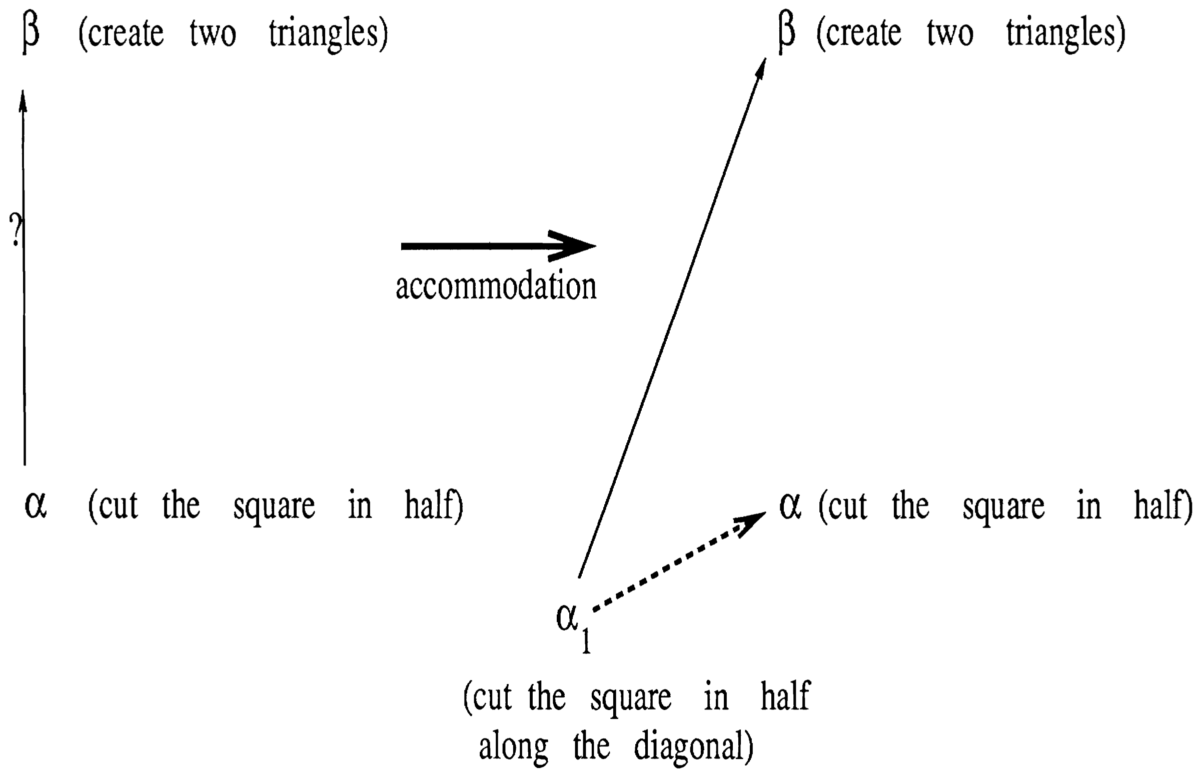
β (create two triangles)
?
α (cut the square in half)
accommodation
β (create two triangles)
α (cut the square in half)
α1
(cut the square in half along the diagonal)

Figure 2.1: Schematic depiction of the first kind of accommodation

In Ex. 6, we have β = *create two triangles*, α = *cut the square in half*, α_1 = *cut the square in half along the diagonal.*

In the second case, it is necessary to find the conditions under which the current instruction makes sense: here, that the urn is in the other room, or at least, that the other room contains some indications as how to get the urn. These conditions can be worked out by relating the effects of *go* to what the performance of *get* requires. Lewis's *getaway car* example is similar, although more complex, as *the getaway car* has to be related to the content of the plan for *stealing plutonium.* Schematically, one could represent this kind of inference as in Fig. 2.2 – β is the goal, α the instruction to accommodate, A_k's the actions belonging to the plan to achieve β, C for the necessary assumptions. Ex. 2 would result in the configuration shown in Fig. 2.3.

In this case the only object that needs to be accommodated is α. However, it may happen that also the goal β needs to be accommodated with respect to the structure of the intentions that H has built so far. In general there may be structures of action descriptions that need to be accommodated, such as those deriving from the interpretation of purpose clauses. However, I will try to keep the process hierarchical, namely, first accommodate the goal β, then the action α. After all, the goal β has more importance: the agent first has to commit to it, and only after that he will commit to a way of achieving it.

It may happen that the two kinds of inference need to be combined, although no example I have found so far seems to require it. In this case, we would need to find the conditions under which an instruction α makes sense in the context of a certain goal, and we may then discover that the relevant action is not α, but a more specific α_1, as in Fig. 2.4.

In the latter case, it may also be necessary to check whether α_1 remains compatible with the higher goal(s).

Kautz. Kautz uses an *is-a* hierarchy and a *part-of* hierarchy to infer what plans a certain event can be part of; he then uses the goals he discovers to constrain the search for the plan of which the next event is *part-of.* Suppose he starts with an event description ϵ; he may then deduce that ϵ can be part of the plans to achieve G_1 or G_2. A second event description ω is in turn part of plans to achieve G_1 or G_3. Finally, by merging the two subgraphs, Kautz obtains a description of what plans both ϵ and ω may be part of - see Fig. 2.5.

The difference between accommodation and Kautz's inferences is that accommodation doesn't assume an exact description of the action to be performed, and in fact uses a known goal to compute the more refined actions needed to satisfy the goal. Kautz goes the other way, as he exploits known and complete descriptions of events to infer higher and unknown goals.

Pollack. Pollack in [1986] is concerned with finding the relation between a known goal G and a known action α, relation that for her is embodied in an *explanatory plan*, or *E-plan.* She finds the proper E-plan relating G and α by searching in parallel for E-plans of which α is part, and for E-plans which achieve G. The important point is that certain conditions have to hold for an E-plan to achieve G. If these conditions don't hold, then the algorithm will search for another E-plan, possibly independent from α, that can achieve G, and will suggest it to the user – see Fig. 2.6.

Therefore, both accommodation and Pollack's inference processes heavily rely on already knowing the goal. However, if α doesn't achieve the goal, my approach is to compute constraints on it, so that α can be understood as α_1, that does achieve G; instead, Pollack finds another E-plan that does achieve G, where α and the new E-plan are not necessarily related.

Given that Pollack determines whether an E-plan does achieve G by means of conditions that have to hold for the relation to go through, she needs to know these conditions a priori to be able to evaluate them in the current situation. In recent work, Lochbaum [1991b] uses a similar representation, and makes it more flexible

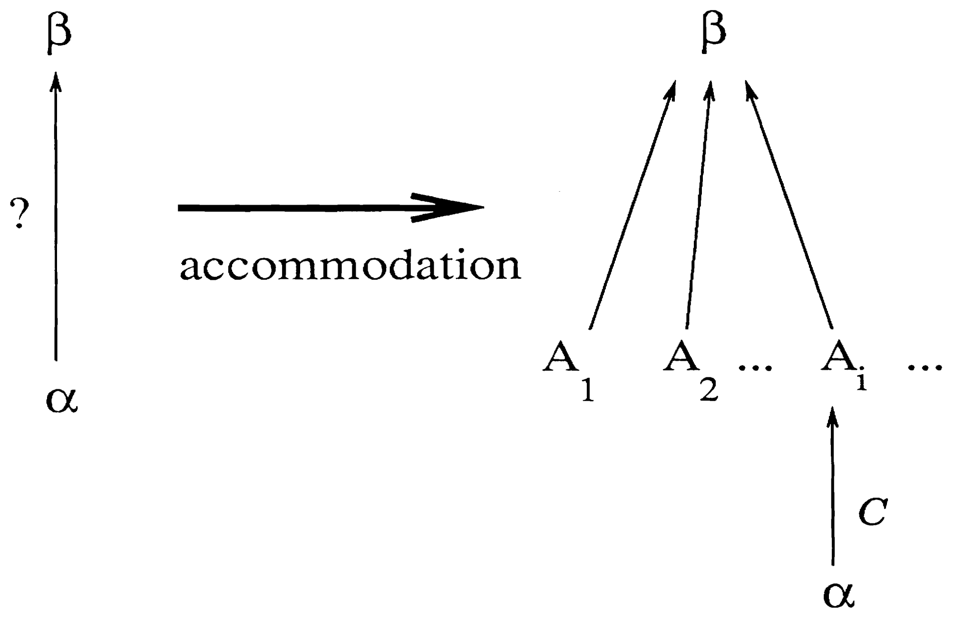

Figure 2.2: Schematic depiction of the second kind of accommodation

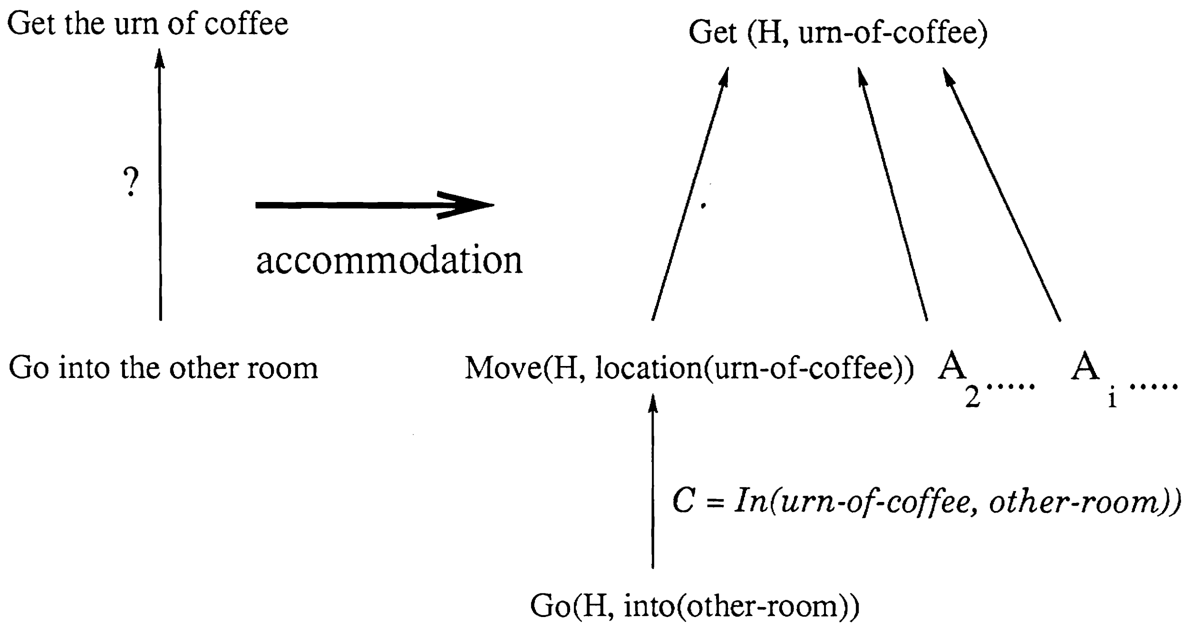

Figure 2.3: Example of the second kind of accommodation

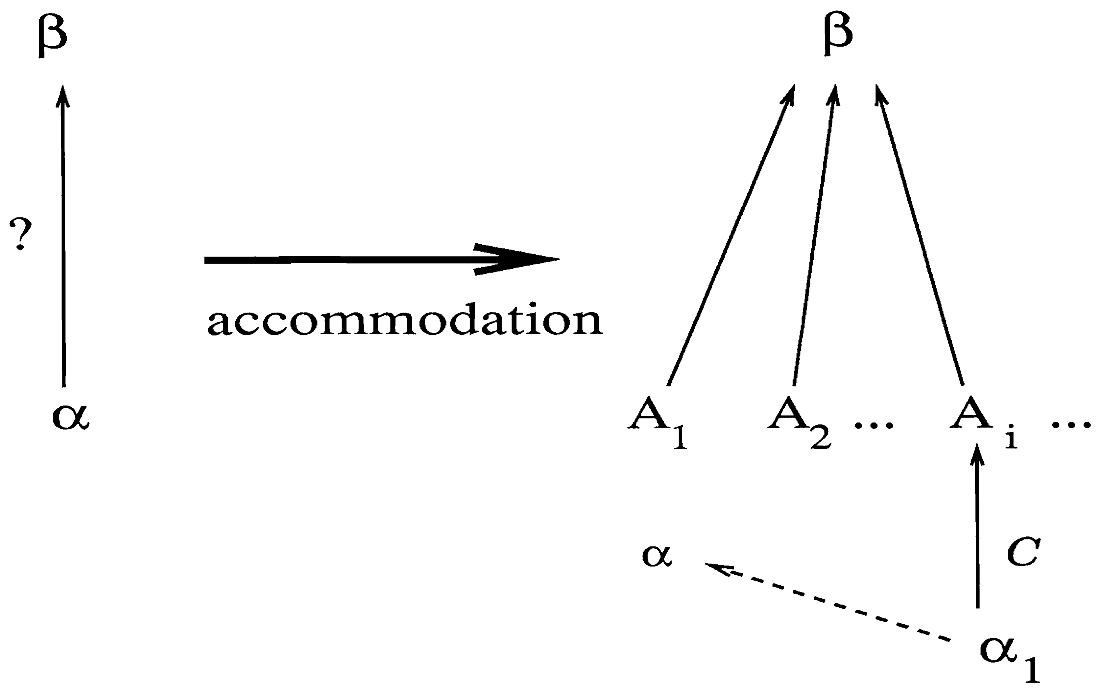

Figure 2.4: The two kinds of accommodation combined

by computing the possible variable bindings under which such conditions may hold, while Pollack has these conditions automatically grounded and simply verifies whether they hold; however, also Lochbaum needs to state all necessary conditions in advance. However, there are other conditions, like the assumption *urn in the other room*, that may be necessary in order for this relation to go through, and with which I am concerned, while they are not.

The distinctions between the inferences I want to account for and the ones accounted for by Kautz and Pollack are important, but in addition there is an underlying assumption that makes accommodation very different from the work done so far under the label *plan recognition.* The point of view that the actions described in the input have to be considered as *linguistic objects* is at the basis of my efforts. I treat the contents of instructions as descriptions, and most likely, incomplete ones. I am interested in explaining how the *real* action descriptions that people use and understand can be accounted for. The need of treating actions as linguistic objects is not addressed at all in plan recognition work [8].

2. **Is accommodation different from the approach in [Alterman *et al.*, 1991]?**
The two may be seen as similar, because both advocate adapting H's plan to the instructions. However, I take the view that instructions are continually integrated into the general plan that the agent is building, not only when the execution of an acquired procedure breaks down.

[8]Others have looked at plan recognition in a linguistic setting (e.g. [Lambert and Carberry, 1991], [Litman and Allen, 1990]), but their linguistic considerations concern recognizing which speech-acts a certain utterance is an expression of, not what actions it describes.

$$\begin{tikzcd}
\text{1)} & ?G & & & G_1 & & G_2 \\
 & & {} \arrow[r] & {} & & & \\
 & \varepsilon \arrow[uu] & & & & \varepsilon \arrow[uul] \arrow[uur] &
\end{tikzcd}$$

$$\begin{tikzcd}
\text{2)} & ?G & & & G_1 & & G_3 \\
 & & {} \arrow[r] & {} & & & \\
 & \omega \arrow[uu] & & & & \omega \arrow[uul] \arrow[uur] &
\end{tikzcd}$$

$$\begin{tikzcd}
 & & & & G_1 & \\
1 + 2 & {} \arrow[r] & {} & & & \\
 & & & \varepsilon \arrow[uur] & & \omega \arrow[uul]
\end{tikzcd}$$

Figure 2.5: Kautz's inferences

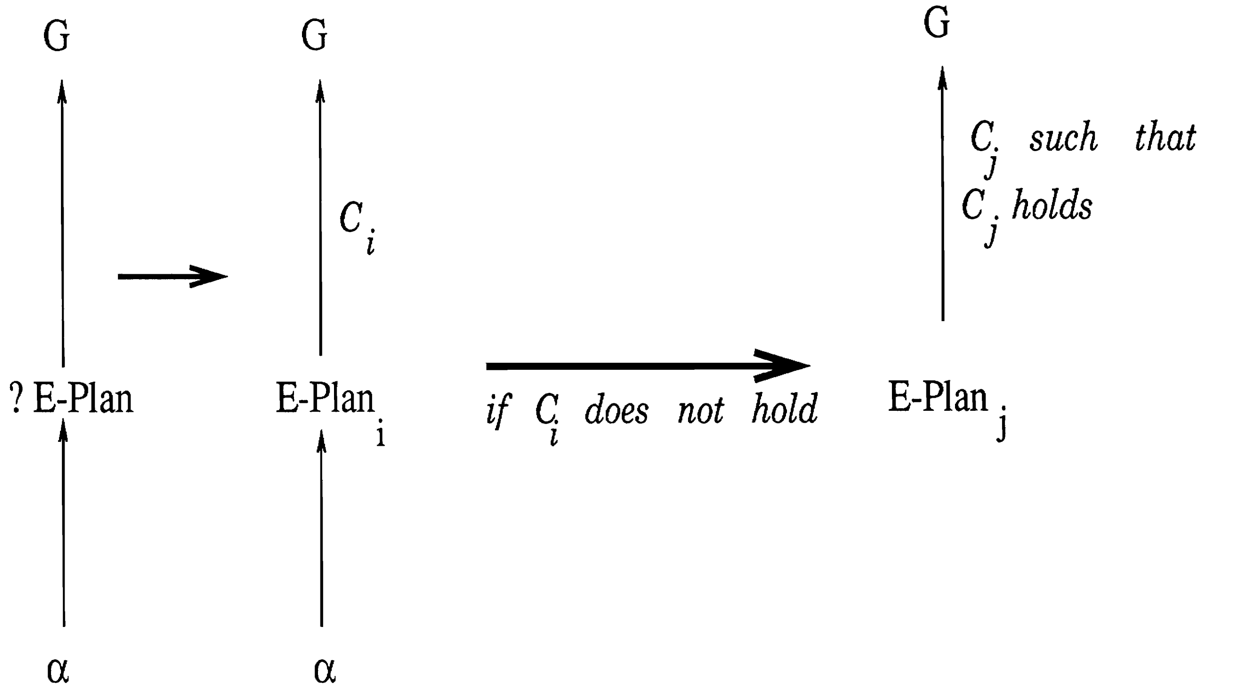

Figure 2.6: Pollack's inferences

To show that instructions are not only used in the way Alterman et al. suggest, consider following directions to reach a certain place in town. This is a typical case in which known actions, such as *go straight until ..., turn left at ..., cross over ...*, are related in a way that teaches the agent something new. However, this does not imply that all that the agent is doing is mapping reference points in the directions onto their referents in the environment and then executing the command. Directions, as all other instructional text, may involve complex imperatives, such as purpose clauses – *to get into the building, turn left at the first intersection and ring the bell at the second door on your right*; negative commands – *turn left, but don't take the first left because it is a blind alley*; warnings – *Take the first left, but be careful if you're driving because it is very steep.*

One can't invoke a *learned* procedure for getting to a certain place if one doesn't know how to get there. Of course following directions may involve breakdowns of the kind [Alterman *et al.*, 1991] describes. If the agent gets lost, he will do something to repair his plan, such as consulting a map or asking a passer-by for further directions. In the latter case, the agent will be given a new set of instructions, that he will have to integrate into the known plan, in a way presumably very similar to the process described in [Alterman *et al.*, 1991]. However, the general process of following directions seems to me much more similar to the one Chapman discusses, namely, putting together some basic building blocks in a novel way.

A problem that [Alterman *et al.*, 1991] accounts for, which I don't, is learning. Following directions to get to a certain place will teach the agent something new, namely, how to get there. One could say that in general it is *following* instructions, rather than *interpreting* them, that teaches the agent something new; this newly acquired knowledge may then be subsequently used as needed. For the moment, I won't address the issue of learning.

2.2 Thesis statement and proposed work

My claim is then that

most instructions don't exactly mirror H's knowledge, but are understood by accommodating them in the context of the general plan H is considering; H's accommodation process is guided by the goal(s) that H is trying to achieve. The concept of goal itself is pervasive in NL instructions, and a NL system which interprets instructions must be able to recognize and/or hypothesize goals, keep track of them, and use them in computing the description of the action to be performed.

- To show that my claim is *justified*:
    1. I will illustrate my views on H's inference processes by means of two different syntactic constructions, *purpose clauses* and *negative imperatives.* I will show what pragmatic functions they perform, and which kinds of inferences H has to do to understand them.
    2. I will show how H's inferences are directed by the notion of *goal.* This is equally true when H interprets *purpose clauses*, that express goals explicitly; or *negative imperatives*, whose correct interpretation is often constrained by the goal in the context of which an action should *not* be performed.
- To show that my claim is *valid*:
    1. granted the importance of *goals*, I will show that, to perform accommodation, H uses
        (a) heuristics derived from the pragmatic usage of certain constructs in instructions;
        (b) knowledge about actions;
        (c) knowledge about relations between actions.
    2. I will then discuss the consequences of such views for a computational model of instructions. I will propose to develop a model composed of
        (a) A speaker / hearer model of complex imperatives based on the one developed in [Cohen and Levesque, 1990b].
        (b) A representation of actions, and of goals, that provides support for performing the inferences I will discuss. I will propose that a hybrid system, á la KRYPTON [Brachman *et al.*, 1983a] should be used. The primitives in the system should be the same that Ray Jackendoff proposes in [1990] for the semantic representation of verbs and actions. Jackendoff's representation will then provide semantic theoretical foundations, while the hybrid system will provide an organization for the lexicon.
        (c) Inference mechanisms that contribute to building the structure of the intentions – the *plan graph* – that H develops while interpreting S's instructions; the representation of such plan graph will also be discussed.

And finally, a comment and a disclaimer.

In the statement of my claim above, I used the word *mirror*. It is obvious that instructions *don't exactly mirror H's knowledge*: if they did, there would be no point in giving instructions in the first place! As I already pointed out, the new information that instructions provide mainly resides in the way that actions known to H are linked together to form a previously unknown plan to achieve a certain goal. However, the way that known actions are described in instructions cannot be expected to *exactly* correspond to the knowledge that H has of those actions: this is a point that has often been overlooked in many systems that deal with instructions, in which a representation such as *cut.square.in-half.along-diagonal* is supposed to capture all the infinite ways in which such an action could be described in an instruction. This is the *inexact match* between input instructions and H's knowledge I am referring to.

As far as regards the disclaimer: the instructions I am dealing with are in Natural Language. Understanding them requires the numerous inference processes that Natural Language processing involves, such as pronominal reference, or telling apart distributive and collective readings of plural predicates, just to name two of them. Clearly I am not going to solve the general problem of understanding language: all I want to show is that, in dealing with instructions, first, the concept of *goal* is necessary; and second, that there are specific inference processes that help make sense of instructions, exploiting, among other things, heuristics derived from particular syntactic structures. I will assume that I can abstract away from issues such as parsing, pronominal and definite reference, the treatment of plurals: such assumptions are even more reasonable because my work is grounded in the project *Animation from Natural Language instructions* [Badler *et al.*, 1990; Webber *et al.*, 1991], where modules devoted to parsing and to representing and updating the discourse model do exist. I will discuss the project in sect. 5.5.

Chapter 3

Purpose clauses

In the previous chapter, I emphasized the role that *goals* play in the *accommodation* process, where *accommodation* is a general name for many of the inferences that an agent performs upon being given NL instructions. I also claimed that such inferences are exemplified in a particularly clear way when the goal is explicitly expressed, as in the case of *purpose clauses.*

In fact, it is *purpose clauses* that motivated my interest in the kind of inferences I am trying to characterize. In this chapter, I will illustrate how accommodation processes come into play in the processing of purpose clauses.

In the following, after a brief explanation of what I mean by the term *purpose clauses*, I will discuss:

1. what entities the matrix clause and its adjoined purpose clause describe;

2. what functions purpose clauses perform;

3. what kinds of relations between actions purpose clauses embody.

I will conclude with some brief remarks about the consequences of such analysis on a computational model of instructions, including a speaker / hearer model, action representation, and inference processes. I will go into the relative technical proposals in ch. 5.

My observations are based on one hundred and one consecutive instances of naturally occurring purpose clauses, which I collected from a how-to-do book on installing wall coverings, and from two craft magazines [1].

3.1 Purpose clauses and action descriptions

At a very general level, purpose clauses can be characterized as follows:

1. S uses them to explain to H the goal β to whose achievement the execution of α contributes.

[1]To be precise, 26 pages of the book provided 51 examples; all the remaining 50 examples come from the "project sections" of the magazines.

2. They generally relate action descriptions at different levels of abstraction, such as a physical action and an abstract process, as in Ex. 8 below, or two physical actions, but at different levels of granularity, as in

   Ex. 7 *Bend your knees to lift a very heavy object.*

3. As I already observed in the previous chapter, not only does the goal β explain to H why he should do α, it also constrains the interpretation of α – see Exs. 5 and 6. Therefore, from S's point of view, we can look at purpose clauses as constraints on the action to be executed; accordingly, H will have to compute the action to execute.

Before going into a more detailed analysis of purpose clauses, I'd like to make some brief remarks on my use of terminology. I am using the term *purpose clauses* to refer to clauses that express the agent's purpose in performing a certain action. I have concentrated on infinitival *to* constructions, but other clauses, such as those introduced by *so that, such that* may also perform the same function. Therefore, I am not using the term *purpose clause* in the technical way it has been used in syntax; there, it refers to a particular type of infinitival *to* clauses, which are adjoined to NPs. In contrast, the infinitival clauses I have concentrated on are adjoined to a matrix clause, and are termed *rational clauses* in syntax; in fact all the data I will discuss in this chapter belong to a particular subclass of such clauses, subject-gap rational clauses. For a syntactic account of rational and purpose clauses and their differences, see [Jones, 1985], [Hegarty, 1990].

In the following, I will discuss the following dimensions of analysis:

1. what is described by the matrix and the purpose clauses – namely, actions or states;

2. what kinds of goals are expressed by a purpose clause;

3. what kinds of relations between actions are expressed by purpose clauses.

3.2 Only action descriptions in purpose clauses?

So far, I have been implying that both matrix and purpose clause describe an action, α and β respectively:

Ex. 8 To mix light colors, *place white in the dish first and add color slowly until desired color is reached.*

However, there are rare cases in which one of the two clauses describes a state σ, while the other clause describes an action. I actually found only one such case, in which the matrix clause describes the state σ:

Ex. 9 *To be successfully covered,* **a wood wall must be flat and smooth.**

I haven't found any instances in which both matrix and purpose clause describe a state. Intuitively, this makes sense because what the speaker wants to express here is the purpose why a certain action should be performed; even in the syntactic literature this function is recognized, e.g. [Hegarty, 1990] says that *rational clauses denote a rationale on the part of the agent to carry out the action described in the matrix clause* [2].

3.3 Which kinds of goals?

I have been saying that a purpose clause expresses a goal to which the action described in the main clause contributes. As I said, the goal β gives a more abstract description of the action α that the main clause describes, or gives a more general context for the execution of α. In most cases, the goal β describes a change in the world. However, in some cases

1. The change is not in the world, but in H's knowledge. By executing α, H can change the state of his knowledge with respect to a certain proposition or to the value of a certain entity.

   Ex. 10 **To determine if a curve is too small for the blade,** *use this rule of thumb: the smallest circle you can accurately cut will have a radius twice the width of the blade you're using.*

   Ex. 11 *You may want to hang a coordinating border around the room at the top of the walls.* **To determine the amount of border,** *measure the width (in feet) of all walls to be covered and divide by three. Since borders are sold by the yard, this will give you the number of yards needed.*

   Many of such examples involve verbs such as *check, make sure* etc. followed by a *that*-complement describing a state ϕ. The usage of such verbs has the pragmatic effect that not only does H check whether ϕ holds, but, if ϕ doesn't hold, s/he will also do something so that ϕ comes to hold.

   Ex. 12 *To attach the wires to the new switch, use the paper clip to move the springtype clip aside and slip the wire into place. Tug gently on each wire* **to make sure it's secure.**

2. The purpose clause may inform H that the world should *not* change, namely, that a given event should be prevented from happening:

   Ex. 13 *Tape raw edges of fabric* **to prevent threads from raveling as you work.**

   Ex. 14 **To prevent weeds from getting a foothold in the garden,** *mulch your seedlings when they are several inches high. Use a thick layer of straw, shredded bask, ground corn cobs, cocoa bean hulls, leaves, or newspaper. The mulch will smother weeds and increase soil moisture.*

From a discourse processing point of view, interpreting a purpose clause may affect the discourse model, in particular by introducing new referents. This happens when the effect of α is to create a new object, and β identifies it. Verbs frequently used in this context are *create, make, form* etc.

[2] There are clearly other ways of describing that a state is the goal of a certain action, for example by means of *so/such that,* but I won't deal with such data.

Ex. 15 *Join the short ends of the hat band* **to form a circle.**

Similarly, in Ex. 6 the discourse referents for the triangles created by cutting the square in half, and in Ex. 11 the referent for *amount of border* are introduced. Ex. 11 shows that the function of introducing a new referent, and the previous functions I mentioned, are not mutually exclusive.

3.4 Which relations between actions?

So far, I have talked about α *contributing* to achieve the goal β. The notion of *contribution* can be made more specific by examining naturally occurring purpose clauses. In the majority of cases, they express *generation*, and in the rest *enablement*. In the following, I will define both relations, I will provide examples of both, and finally I will talk about the fact that there is no clear cut distinction between the two.

3.4.1 Generation

Generation is a relation between actions that has been extensively studied in the literature, first in philosophy [Goldman, 1970] and then in planning [Allen, 1984], [Pollack, 1986], [Balkanski, 1990], [Lochbaum, 1991a].
Informally, if action α generates action β, we can say that β is executed **by** executing α. Examples are (from [Goldman, 1970]):

> *The agent G turns on the light* **by** *flipping the switch*, or
> *G checkmates his opponent* **by** *moving his queen to king-knight-seven.*

Without going into too many details, we can say that an action α generates another action β iff:

1. α and β are simultaneous;
2. α is not part of doing β (as in the case of *playing a C note* as part of *playing a C triad* on a piano);
3. when α occurs, a set of conditions C hold, such that the joint occurrence of α and C imply the occurrence of β. For example in the *turning on the light* example, C may include that the wire, the switch and the bulb are working.

Although there is no generation relation between α and β if α is part of a sequence of actions $\mathcal{A}$ to do β, there maybe a generation relation between $\mathcal{A}$ and β, if the conditions above hold on $\mathcal{A}$ and β.

Generation is a pervasive relation between action descriptions in naturally occurring data. There are cases in which it is expressed by the preposition *by*, as in:

Ex. 16 *Remove excess paste* **by wiping gently or blotting with a damp sponge.**

It can also be expressed with a simple free adjunct, as in:

Ex. 17 *As you work, clean the surface thoroughly each time you change grits,* **vacuuming off all the dust and wiping the wood with a rag dampened with turpentine or paint thinner.**

However, *by* clauses are not as common as purpose clauses: I found only 27 of them in the same corpus. Similarly, adjuncts are not often used to express generation: only 10 out of 97 adjuncts found in a corpus that includes the one used for purpose clauses express generation [Webber and Di Eugenio, 1990]. It looks like generation in instructional text is mainly expressed by means of purpose clauses. They may express either a direct generation relation between α and β, or an indirect generation relation between α and β, where by *indirect generation* I mean that α belongs to a sequence of actions $\mathcal{A}$ which generates β.

As an example of direct generation, consider Ex. 6, repeated here for convenience:

Ex. 18 *Cut the square in half* **to create two triangles.**

In this case, the only action that needs to be done *to create two triangles* is *cutting the square in half [along the diagonal].*

As an example of indirect generation, consider Ex. 8, or

Ex. 19 To cut parts F [3] *measure 1-3/4" from the top and mark. Measure from that point 7-1/2" to the left and mark. Measure up from the bottom 1-3/4" and mark. Draw a line from the first mark to the third mark to create the slant for the sides. Cut along this edge.*

The whole paragraph is an explanation of how *to cut parts F.* It is the whole sequence of actions described in the paragraph that achieves this result. Therefore, the relation between *to cut parts F* and *measure 1-3/4" from the top and mark* is one that I call of *indirect generation*: the latter action description is part of a sequence of actions that generates the former. Notice that the paragraph above contains another purpose clause, *to create the slant for the sides,* which individuates a subgoal in the context of *cutting parts F.*

3.4.2 Enablement

Following first Pollack [1986] and then Balkanski [1990], *enablement* can be defined as holding between two actions α and β if and only if an occurrence of α brings about a set of conditions that are necessary (but not necessarily sufficient) for the subsequent performance of β.

An example is *buying ingredients* **enables** *preparing the dish* (from [Balkanski, 1990]). In this case, buying ingredients brings about a condition that is necessary for the dish preparation action to be *executable.*

There are not many purpose clauses that express an enablement relation between α and β. In fact, there is only one example that truly seems to express enablement:

Ex. 20 *Unscrew the protective plate* **to expose the box.**

Unscrew the protective plate enables *taking the plate off* which generates *exposing the box.*

Another case that at first sight could be classified as expressing enablement, but that actually doesn't exactly correspond to the notion of enablement is:

[3] *Parts F* refers to a diagram.

Ex. 21 *Build this simple but elegant corner shelf* **to display all the other decorative pieces found in this issue.**

Having the shelf is not a necessary condition for *displaying the decorative pieces.* In fact, it's not even the case that H has to adopt the intention of displaying such pieces; probably such an example should be classified as *facilitation* – see [Balkanski, 1990].

3.4.3 Generation and enablement in modeling actions

That purpose clauses express generation and enablement is a welcome finding, given that these two relations have been proposed by several researchers – [Allen, 1984], [Pollack, 1986] – as necessary to model actions.

Pollack in her thesis [1986] is particularly vocal in advocating these two relations as the basis for action representation. Her motivations stem from the need for a perspicuous representation for an agent's plans, representation that can support her view of plans as *mental phenomenon.* She notices that in the plan inference literature plans are generally derived by composing basic planning operators, which are represented as follows:

```
       Header:  β
  Preconditions :  P
        Body :  α
```

She criticizes such a representation because the relation between α and β is not precisely defined: researchers have assumed any of *causes, is-a-precondition-of, is-a-way-to.* She then prefers to use a representation based on *generation* and *enablement*, which she can precisely define.

I would like to add two further motivations for using *generation* and *enablement.* One is the presence of such relations in Natural Language instructions. The other is that using such relations to model actions allows us to draw conclusions about action execution as well–a particularly useful consequence given that my work is taking place in the framework of the *Animation from Natural Language* project [Badler *et al.*, 1990; Levison, 1991; Webber *et al.*, 1991] in which the input instructions do have to be executed, namely, animated.

Knowing that two actions are related by generation allows us to draw the conclusion that, while two actions are described, only α, the generator, needs to be performed. For example, in Ex. 18, there is no *creating* action per se that has to be executed: the physical action to be performed is *cutting.* As already pointed out in the previous chapter, the fact that only the generator α has to be executed does not mean that qualifications to the description of the generatee β should be disregarded: in fact we know that the goal β *constrains* the interpretation and / or execution of α. Consider Ex. 17: the instruction to the agent is that s/he has to *vacuum* and *wipe* every time s/he changes grits, which is a qualification to the description of *cleaning.*

In contrast to generation, if α enables β, the agent knows that, after executing α, β still needs to be executed. In fact, if α enables β, α has to temporally precede β, where temporal precedence should be understood as α beginning, but not necessarily ending, before β: in Ex. 22, *hold* has to continue for the whole duration of the *fill.*

Ex. 22 *Hold the cup under the spigot to fill it with coffee.*

In addition, in the same way that the generatee affects the execution of the generator, so the enabled action affects the execution of the enabling action. Consider, for instance, the difference in the interpretation of *to* in *go to the mirror*, depending upon whether the action to be enabled is *seeing oneself* or *carrying the mirror somewhere else*.

Having established that these two relations are useful to model Natural Language descriptions of actions, the problem is to formalize them adequately.

One issue that has to be addressed is the fact that the distinction between generation and enablement becomes blurred when we extend the concept of generation to a sequence of actions $\mathcal{A} < \alpha_1, \alpha_2, ..., \alpha_n >$ that generate another action β [4]. What is then the relation between α_i and β? I will suggest that if $< \alpha_1, \alpha_2, ..., \alpha_n >$ generates β, *enablement* may relate any pair α_i and α_k, while the relation between α_i and β can be termed *indirect generation*. Actually, at least for $i = 1$, the relation between α_i and β could be equivalently seen as enablement.

I will go back to the difference between *generation* and *enablement* in chapter 5.

3.5 Consequences for a computational model of instructions

I see three kinds of consequences that purpose clauses have on computational models of instructions.

First of all, they require that existing computational models of intentions, such as Cohen and Levesque's, be augmented to take into account the particular interpretation process that an imperative containing a purpose clause requires of the hearer – in particular, that *Do* α *to do* β is used to tell H that s/he should adopt the intention to do β; and that α contributes to the execution of β. It seems that the goal β has more importance than α: namely, we should account for the fact that a (cooperative) hearer has to commit to β, but could choose another α to achieve β. That the goal β has more importance than α is shown also when the purpose clause is adjoined to a negative matrix clause, as in *Don't use chemicals to clean your parquet floor*. Needless to say, the hearer has to adopt the intention of doing β, *cleaning the parquet floor*, even if he must not adopt the intention of doing α, *use chemicals*.
I will sketch a first formalization of these notions, based on [Cohen and Levesque, 1990b], in chap. 5.

Second, dealing with purpose clauses, and with naturally occurring action descriptions in general, raises interesting issues with respect to the action formalism that we should employ. Some preliminary conclusions are that we should be able to represent actions at different levels of specificity, and that generation and enablement should be included among the relations holding between actions.

Third, the abundance of purpose clauses in naturally occurring instructions can help focus the accommodation process, both in suggesting the kind of inferences that need to be done, and in offering a way to direct such inferences.

In sect. 2.1.3 I sketched two kinds of accommodation that directly derive from purpose clauses, namely, inferring the assumptions under which a certain instruction makes sense, and deriving a more specific description of the action to be executed, by applying to it the constraints deriving from the goal.

[4]Goldman himself allows for sequences of actions to generate another action – see [Goldman, 1970].

The solution I have adopted so far to deal with the latter kind of inference is to use hybrid knowledge representation systems [Brachman *et al.*, 1990]. However, such representation lacks a well-defined set of semantic primitives, necessary to provide a sound lexical decomposition for verbs. I found a source for such semantic primitives in Jackendoff's conceptual structures [Jackendoff, 1990]. Part of my proposal will then be to integrate the two frameworks together.

All these issues will be discussed in more detail in chap. 5.

Chapter 4

Negative imperatives

In this chapter, I will consider the following two questions:

1. Negative imperatives are not limited to *don't do α*. Are different kinds of negative imperatives used exactly in the same way, or does their distribution tell us something on the expectations that S has on the intentions H may adopt?

2. What inference processes does H perform to understand negative imperatives? In particular, what role – if any – does the concept of *goal* play? Consider:

   Ex. 23 *Caring for the floor. A good paste wax – not a water-based wax - will give added protection to the wood. Buff about twice a year; wax about once a year. Excessive waxing can cause wax to build up, detracting from the floor appearance.*
   Dust-mop or vacuum your parquet floor as you would carpeting. **Do not scrub or wet-mop the parquet.** *Use a damp cloth to remove fresh food spills.*

   H will understand the negative imperative by recognizing that *dust-mop, vacuum, scrub, wet-mop*, are all possible candidates for achieving the goal of *cleaning the parquet*; and that, although *scrub* and *wet-mop* may in general be performed to achieve a goal such as *cleaning floor*, they are not a viable alternative in this case.

   The question is then, what kind of relations the actions described in negative imperatives bear to one another and to a possible more general goal that has already been established – I am talking about *already established goal* because all the examples I found are interpreted with respect to the previous, and not the following, text. Making sure that this generalization really holds is part of future work.

As we will see, the two questions are not independent, namely, different kinds of negative imperatives in general perform different functions, and concern actions that bear different relations to one another.

In the following, I will classify the data into two general classes, on the basis of the expectations that S has on the intentions that H may or may not adopt; I will then discuss some consequences of the analysis of negative imperatives for a model of the inferences H has to make.

4.1 The data

There seem to be two basic classes of negative imperatives [1]: negative imperatives proper, characterized either by the negative auxiliary *don't* [2], or by negative polarity items, such as *never* and *nothing*; the other class is formed by verbs such as *take care, be sure* and the like – *TC verbs* for short – followed by a negative infinitival complement, such as *"not to ..."*, or *"to ... negative polarity item ..."*. I will call the former class of imperatives the *DONT* type and the latter the *neg-TC* type [3].

The data is distributed as follows:

1. 36 instances of the *DONT* type, and in particular
   (a) 32 with *don't* or *do not*:

   Ex. 24 *Your sheet vinyl floor may be vinyl asbestos, which is no longer on the market.* **Don't sand it or tear it up** *because this will put dangerous asbestos fibers into the air. It must be covered over.*

   (b) 4 with *never*

   Ex. 25 **Never mix cleaners containing acid or ammonia with chlorine bleach.** *The chemical reaction releases the chlorine as a poisonous gas.*

2. 29 examples of the *neg-TC* type:
   (a) 4 with *take care*

   Ex. 26 *To book the strip, fold the bottom third or more of the strip over the middle of the panel, pasted sides together,* **taking care not to crease the wallpaper sharply at the fold.**

   (b) 4 with *be sure* or *make sure*

   Ex. 27 *To wash, load clothes into tub,* **making sure not to overfill it.**

   (c) 22 with *be careful*

   Ex. 28 *If your plans call for replacing the wood base molding with vinyl cove molding,* **be careful not to damage the walls** *as you remove the wood base.*

4.2 Different uses for different negative imperatives

As far as semantics goes, it is clear that a *DONT* imperative could be used when a *neg-TC* one is used: an expression like *take care not to do* α entails *don't do* α [4]. In fact, in terse

[1] I collected data from two "how-to-do" books, both from the Sunset publication company – *Wall Coverings* and *Tile* – plus a few from detergent and toiletry containers.

[2] I am not distinguishing between *don't* and *do not.*

[3] Negative imperatives can also be formed by means of negated modals, such as *shouldn't.* Presumably such instances could be reduced to *DONT* imperatives, although this characterization doesn't seem to be totally appropriate. Given that I found only one such example, I won't consider them in the following.

[4] It is not clear what *entails* really means here, given that instructions are difficult to model in the usual model-theoretic semantics. I am using this term in an intuitive sense.

instructions only *DONT* imperatives are found. However, the instructions I have looked at come from instructional text, and there the two classes of negative imperatives are indeed used in different ways, that depend on the expectations S has on the intentions H will adopt – notice that in this proposal I will not address the problem of *why* S has such expectations. The two classes of negative imperatives pattern as follows:

DONT imperatives. *Don't do α* appears to be used when S thinks that H is likely to adopt the intention to perform α, and wants to prevent H from adopting it.

That there is a correlation between S's expectation about the intentions which H is likely to adopt, the use of a *DONT* imperative, and the real likelihood that H adopts those intentions is shown by an infelicitous usage of a – really occurring! – *DONT* imperative:

Ex. 29 *# If you must replace a tile, first cut around the edges with a circular saw. Set the blade to the depth of the tile and* **don't damage adjoining tiles**.

Damaging x is hardly an intention that an agent would normally adopt, so that using a *DONT* imperative results in an infelicitous utterance: I will come back to this example in sect. 4.2.3.

Neg-TC imperatives. A form like *"Do β. Take care not to do α"* appears to be used when S wants H to perform a certain β and thinks that H may adopt the intention of performing it in an undesirable way, or that H may NOT adopt the intention of preventing an undesirable consequence of β – α describes such an undesirable feature.

It appears that in general lexical items such as *take care* draw H's attention to something he may otherwise overlook, both in negative and in positive imperatives:

Ex. 30 *For paint spots on arms and hands, rub lightly with thinner, working quickly. On face and neck, dab the spots off with a cloth dipped in thinner.* **Be extremely careful to keep the thinner away from your eyes.**

As when TC verbs are used in negative imperatives, we have an action β *"dab the spots off with a cloth dipped in thinner"* that has to be performed under the additional constraint α *"keeping the thinner away from your eyes"*.

However, it appears that when TC verbs are used in negative imperatives there always is a β in discourse which α is meant to constrain; on the other hand, a preliminary analysis of occurrences of TC verbs in positive imperatives shows that they can also be used in the absence of such β:

Ex. 31 *Be sure to read the adhesive manufacturer's instructions carefully.*

In this case, *be sure* seems to be used only for added emphasis, to stress the importance of *reading the adhesive manufacturer's instructions carefully.*.

My conclusion, which I will support with further evidence in the rest of this chapter, is that *neg-TC* imperatives express more than simple negation. A thorough analysis of TC verbs used in positive imperatives will be instrumental in analyzing their behavior in negative imperatives too.

I will now justify the different characterizations proposed for *DONT* and *neg-TC* imperatives; in sect. 4.3 I will provide more precise specifications.

4.2.1 DONT imperatives

The expression I will examine in this section is *Don't do α*, which may be used by S to convey to H that

α **is an undesirable alternative** to a β that S tells H to do. α may come to H's mind by analogy, as an *alternative* to some other β that S suggests, as in Ex. 23, whose relevant part is repeated here again for convenience:

> **Ex. 32** *Caring for the floor. A good paste wax – not a water-based wax - will give added protection to the wood. ... Dust-mop or vacuum your parquet floor as you would carpeting.* **Do not scrub or wet-mop the parquet.**

> As I stated in the introduction to this chapter, the goal to be achieved is γ *"cleaning the parquet"*. Notice that this is not stated explicitly, but requires a step of plan recognition: this step will probably take advantage of the fact that the whole paragraph is under the heading *Caring for the floor*, which is presented as subdivided into *waxing* and *cleaning*. S suggests to H two methods to achieve this goal, and then prevents H from thinking that other apparently equivalent methods could be used.

> Ex. 32 is particularly "clean", in that the fact that *scrub* and *wet-mop* are (undesirable) alternatives with respect to *dust-mop* and *vacuum* can be easily inferred

> 1. either by using an action inheritance hierarchy and by modeling all these actions as daughters of the same parent, *clean floor*;
> 2. or by modeling them as different methods to achieve the same goal *clean floor*, and reasoning "backwards" from the goal to understand what all these actions have in common.

> In other cases, the knowledge that may suggest α to H as an alternative to β is more complex:

> **Ex. 33** *Step 4. Lift brush straight up, letting excess paint drip back into pail. Gently slap both sides of brush against inside of pail two or three times.* **Don't wipe brush across lip of pail**, *or bristles may separate into clumps, leaving less paint on brush.*

> *Slap* and *wipe* can be seen as alternative ways of getting rid of excess paint on the brush – in addition to letting it drip, which is a slow way of accomplishing such goal. However, the fact that *wipe* is an alternative to *slap* and that S feels the need to caution H against it comes from general knowledge about what people do with brushes loaded with an excessive amount of some kind of viscous substance.

> Notice the importance of the goal γ in understanding Exs. 32 and 33: in the former the goal is *cleaning the parquet*, in the latter *getting rid of excessive paint from the brush.* Understanding that the relation between the actions to be performed – e.g. *vacuum* – and the undesirable ones – e.g. *scrub* – is one of alternative requires understanding the common goal that they all achieve.

> Therefore, *DONT* imperatives provide more evidence about the necessary role that goals play in understanding the relations between the described actions.

α **may state a general goal,** independent from other actions in the discourse. S tells H not to adopt an intention relative to some general tasks. The actions not to be performed are independent from other actions in the discourse. Of course, they may be related to other more general goals, that either have already been established, such as *Hang wallpaper* in Ex. 34 below, or that are generally held as desirable, such as *Don't put anybody's health in jeopardy*, which underlies Ex. 24 above. Consider:

Ex. 34 *If you're using a lightweight or porous wallpaper, profit from these "don'ts":*
a. **Don't hang the wallpaper over dark painted walls***; it could show through. Lighten the walls with a flat, oil-base enamel undercoat before hanging the wallpaper.*
b. **Don't paper over a dark colored or patterned wall covering.** *First, lighten the background with paint or lining paper.*
c. **Don't hang the wallpaper over an existing wallpaper** *if the ink from the existing covering comes off; it could bleed through the new covering. To test, moisten a small piece of the covering with a clean sponge. If the ink comes off, seal the old covering first.*

Other. *Don't do* α can be used to tell H that an action β should not be followed by α, in particular if α affects the outcome of β. Consider:

Ex. 35 *Use a vinyl-to-vinyl paste for any type of border you plan to hang over wallpaper. To paste the border for hanging, cover the entire back with paste and book the strip;* **don't crease the folds.**

The action *booking the strip* creates folds; the action of *creasing the folds* is seen as independent from *booking*, and H is told not to perform it.

There are few additional cases in which a *DONT* imperative is used to caution H against possible side-effects of a certain action, or against possible undesirable ways of executing it, or to give termination conditions for it:

Ex. 36 *The tiles will cut more easily if warmed in sunlight or over a furnace vent.* **Don't overheat** *or the tiles may scorch or melt.*

Ex. 37 *After washing the tile, rinse it thoroughly to remove detergent film; then wipe with a soft, dry cloth. For stubborn dirt, scrub tiles with a white, cleansing powder.* **Don't use a cleaner containing bleach***; this can pull the color out of the colored grouts.*

Actually the generalization that *DONT* imperatives are used when S thinks H may adopt an intention to do α does not seem to hold for the last two examples. For example, in Ex. 37, rather than consciously adopt the intention of using such a cleaner, it is more likely that H may overlook the fact that the cleaner he chooses contains bleach: as we will see, these are typical cases in which a *neg-TC* verb is used.

I will come back to these example in sect. 4.2.3.

4.2.2 Neg-TC imperatives

Neg-TC imperatives caution H against either undesirable ways of performing the action β that S intends H to perform, or undesirable consequences of β. That the negated action α is strictly linked to β is also shown by the fact that many instances of *neg-TC* imperatives are adjuncts to the matrix clause describing β, resulting in patterns such as *"Do β, taking care not to do α"*.

Given an imperative *TC-verb not to do α*, α may be:

An undesirable way of performing β. The description of β is always underspecified, and therefore H has many degrees of freedom in executing it. Consider:

> **Ex. 38** *To make a piercing cut, first drill a hole in the waste stock on the interior of the pattern. The diameter of the hole must be larger than the width of the blade. If you want to save the waste stock for later use, drill the hole near a corner in the pattern.* **Be careful not to drill through the pattern line.**
>
> β is *drill a hole near a corner in the pattern.* The interpretation of *near* still leaves H some choices as regards the exact position where to drill: S constrains them by saying *Be careful not to drill through the pattern line*, having already warned H that the hole must be larger than the width of the blade.

An undesirable effect of β.

1. The undesirable effect may spring from the manner in which H performs a certain action, and be entirely under H's control:

   Ex. 39 *Make a small 1/4-inch slit in the center of each area to be stuffed.* **Be careful not to snip the surface fabric.**

   Another very common way of expressing a (generic) side-effect to be avoided is by means of verbs like *ruin, damage, mar* etc:

   Ex. 40 *When nailing the panels,* **be careful not to mar the surfaces.**

   Ex. 41 *If your plans call for replacing the wood base molding with vinyl cove molding,* **be careful not to damage the walls** *as you remove the wood base.*

2. The undesirable consequence of β may depend on external laws not under H's control: the only thing H can do is prevent such events from happening.

   Ex. 42 *To hang the border, begin at the least conspicuous corner. The work will go much faster if you have someone hold the folded section while you apply the border to the wall.* **Take care not to drip paste onto the wall** *and be sure to remove excess paste.*

   Gravity governs the dripping of paste: H has to perform β in such a way that α does not happen. Notice that the agent can perceive α while doing β. In situations where the agent can't perceive α, it would be more felicitous to say *check that α doesn't happen*.

An explicit culmination point for actions that describe monotonic processes:

Ex. 43 *To wash, load clothes into tub,* **making sure not to overfill it.**

Load clothes into tub has a default culmination condition, which here is reinforced by the explicit *making sure not to overfill it.*

4.2.3 Further evidence for the two classes of negative imperatives

If the two types of negative imperatives do pattern in the way I illustrated, even if such pragmatic functions are not completely clear cut, we should expect to find some cases in which the usage of one instead of the other results in an infelicitous utterance. As I mentioned in sect. 4.2, I did find such an example in my data, repeated here again for convenience:

Ex. 44 *# If you must replace a tile, first cut around the edges with a circular saw. Set the blade to the depth of the tile and* **don't damage adjoining tiles.**

As I said, S uses *DONT* imperatives when she thinks H is likely to adopt the intention to perform α, unless explicitly told not to. However, *damage* is hardly an intention that a person adopts in the context of building or repairing an artifact: rather, it is an undesirable side-effect that can result from a careless execution of α.

I conducted an informal experiment to test whether my intuitions are correct, and whether using *Take care not to damage adjoining tiles* would be more felicitous. My informants agreed with my judgement, and pointed out that the presence of the conjunction is another source of infelicity. In fact, conjunction should be used to relate two conjuncts that have the same function in discourse, but here the two conjuncts are, respectively, a subpart of the action of *cutting the tile*, and a possible negative consequence of such an action. However, all my informants rated conjunction as having a less conspicuous effect than using *don't* instead of *take care not to.*

A last remark on the difference in the usage of these two classes of negative imperatives. That they achieve different results holds even for some apparent counterexamples, in which the two forms of negative imperatives seem to be used interchangeably. Consider the following pair – the first member is Ex. 35, repeated here for convenience:

Ex. 45 a. *Use a vinyl-to-vinyl paste for any type of border you plan to hang over wallpaper. To paste the border for hanging, cover the entire back with paste and book the strip;* **don't crease the folds.**
b. *To book the strip, fold the bottom third or more of the strip over the middle of the panel, pasted sides together,* **taking care not to crease the wallpaper sharply at the fold.**

Although similar, the two examples are not completely equivalent.

In Ex. 45.a, the usage of a DONT imperative shows that the two actions of *booking the strip* and *creasing the folds* are seen as a sequence of separate actions. That is, S tells H to perform *book the strip. Book the strip* creates folds: S intends H not to perform *crease* on the result of the previous action, after such result has been created.

In Ex. 45.b, S explains to H how to *book a strip. Folding* is explicitly mentioned as part of such "recipe"; *creasing the folds* is seen as a possible culmination of *folding*, and therefore as a part of *booking the strips.*

In ch. 5 I show how the Ex. 45.a and .b can be treated – see figs. 5.7 and 5.8.

In sect. 4.2.1, I mentioned that Exs. 37 and 36 are counterexamples to my characterization of *DONT* imperatives. This could be simply due to the fact that the respective functions of *DONT* and *neg-TC* imperatives are not completely clear cut, because, as far as semantics goes, wherever a *neg-TC* imperative is used, a *DONT* imperative could be used; and in fact, *DONT* imperatives are used when the style of the instructions is terse.

As regards Ex. 36 – *Don't overheat [the tiles]* – my feeling is that it could perhaps be accounted for in the same way Ex. 45.a is accounted for: namely, S shows that she is viewing *overheating* as a separate action from *warming*.

4.3 Consequences for a computational model of instructions

A first intuitive characterization of the different effects that *DONT* and *neg-TC* imperatives have on the hearer could be:

DONT: GOAL(H, NOT(perform(H, α)))

neg-TC: GOAL(H, perform(H, β')) $\wedge$ (β' =CONSTRAINED-BY(β, AVOID(α)))

As with purpose clauses, I will work out the details of the speaker / hearer model in the next chapter. An interesting parallel between purpose clauses and *neg-TC* imperatives arises: in the same way that the generator α is constrained by the generatee β contained in the purpose clause, so in many *TC verb not to* α examples, the action β to be performed is constrained by α. Also in this case, therefore, we have to resort to accommodation as the process that accounts for such inferences.

One point that I have not addressed at all so far is *why* S has such expectations on the intentions that H may or may not adopt. Clearly S must have a user model of some sort in mind, but given that instructional text addresses an audience, not an individual, it must be the case that S has certain general expectations of her audience, such as level of expertise etc. To model S's beliefs in such a way that one can predict her expectations is a very interesting topic in itself, a topic that I don't plan to address in the course of my thesis, but which I may touch upon.

As far as the inferences that H has to perform to understand the (possible) relations of α to β, I would first like to distinguish between *internal* and *external* relations between two actions: *internal* relations include α being a specialization, or a side-effect, or a culmination of β; *external* include alternative, generation, enablement [5]. I would characterize *DONT* imperatives as indicating an external and *neg-TC* an internal relation between α and β.

DONT imperatives. If there is a β to which α relates, such a relation will typically be *external*, namely, α and β will not be part of each other, or one a consequence of the other. To understand the relation holding between them, it is often the case that H needs to resort to a third action γ, for example a mutually accepted goal such as *cleaning the parquet* in Ex. 32. A step of plan recognition may be required to infer γ.

[5]The status of temporal relations with respect to being *internal* or *external* is not clear.

As I said, style and conciseness can result in *DONT* imperatives being used instead of *neg-TC* ones; therefore, we could hypothesize that by default H will start by postulating the existence of an external relation between α and β in the presence of a *DONT* imperative. This default can be easily overridden, for example if H knows that the instructions he is dealing with are terse, so that only *DONT* imperatives are used; or if there is a particular lexical item, such as *use*, which, having little meaning by itself, triggers the hypothesis that there is an action to which it can be internally related.

Neg-TC imperatives. The relation between α and β is *internal*. Sometimes the NL surface forms help in identifying such relations more precisely: in Exs. 38 and 39, such inference processes are made easier by the fact that either the same verb, *drill*, or two closely related verbs, *(make a) slit* and *snip*, are respectively used for β and α.

Chapter 5

A computational model of instructions

In this chapter, I will propose a model for understanding instructions, based on the evidence about the accommodation processes and about purpose clauses and negative imperatives I presented in the previous chapters. Such model will account for the initial intentions H develops when he understands an imperative, and for the inference processes that he performs to refine his initial understanding.

Such model will be composed of

1. a speaker / hearer model of imperatives based on [Cohen and Levesque, 1990b], to account for the initial intentions H adopts;
2. an action representation formalism that integrates hybrid knowledge representation systems and Jackendoff's lexical semantic representation, to facilitate accommodation inferences;
3. inference processes that build a structured representation of H's intentions – the *plan graph*.

5.1 A speaker / hearer model of imperatives

Following [Cohen and Levesque, 1990b], I intend to provide axioms on the basis of which the intentions H adopts while interpreting complex imperatives can be accounted for; in particular, I am talking about the intentions that H adopts as an initial "response" to the imperative. Such formalization could be used to model S's beliefs as well, in order to predict the way in which she will phrase her instructions, but this is left for future work.

One may wonder why it is necessary to provide such an axiomatization, given that I will also provide a representation for the structure of H's intentions, and inference processes that build it. Conversely, one may wonder why building the structure of H's intentions is necessary, when through the axiomatization one can account for the intentions H adopts. The answer is that in fact these two parts of the model account for different parts of H's intentions. In particular:

1. Such axiomatization is a declarative representation that links surface form of an utterance and H's intentions. Its first task is to model communication between S and H: therefore, starting with S's goals in uttering a certain kind of imperative, to account for the beliefs and intentions that H will adopt on the basis of such utterance. The second task of such axiomatization is to model what other intentions and beliefs H may adopt on the basis of those he derived from the input utterance [1].

   Such formalization will then allow us to make predictions about the intentions that H will adopt in a more principled way than by triggering procedures. It could also be useful to account for the way a speaker generates a certain kind of imperative, and ultimately, for generating such utterances.

2. Having both the axiomatization and the inference processes that build the plan graph helps to keep distinct the intentions that the agent adopts as a "response" to the input instructions, and those that are adopted as a consequence of accessing the knowledge the agent has about the actions. Keeping them distinct is necessary because accommodation heavily depends on default assumptions. For example, by accommodation we will infer that in *Turn screw to loosen* the real action to be performed is *Turn screw counterclockwise.* This inference depends on the assumption that the screw in question is left-handed. If H later discovers that the screw is actually right-handed, his commitment to *loosening the screw* should be maintained, while the commitment to *turn the screw counterclockwise* should be changed into the commitment to *turn the screw clockwise.* Therefore we must keep track of how these intentions were arrived at.

   To keep things simple, I will model a cooperative agent that will adopt intentions to perform all the actions that S instructs him to do. Clearly, this *obedient* agent is severely limited; as we will see, in Cohen and Levesque's model H adopts an intention α if α is not contrary to his own goals. If in the future I extend the model to deal with more complex agents, this feature will provide a point from where to start.

3. On the other hand, such an axiomatization is not sufficient, and building the plan graph is necessary because the axiomatization only helps to predict which intentions H will adopt, but not to compute the exact objects of such intentions, or to understand how different intentions relate to each other. The plan graph keeps track of such relations; for example, when interpreting sequences of imperatives, the structure of the plan graph built so far helps to understand the next instruction. The axiomatic model only provides for the intentions that H adopts in answer to an imperative, not for how such intentions relate to others that H may have already adopted.

5.1.1 Cohen and Levesque's model

In [1990b], Cohen and Levesque present a model of a cooperative agent. I will first give an overview of the logic at the basis of their model, as they report on it in [1990a] [2].

[1]Pollack in her thesis [1986] presents a formalization of beliefs and intentions that is aimed at the latter task. Although she does address the problem of H modeling S' beliefs underlying a given request, she is not concerned with the task of mapping a given surface form into such beliefs and intentions.

[2]In fact, I will exploit Allen's comments on their formalism in that same book [Allen, 1990].

The logic

Cohen and Levesque develop a modal logic whose model theory is based on a possible-worlds semantics. Their logic has four primary modal operators: BELief, GOAL, HAPPENS (which event happens next), DONE (which event has just occurred).

Time is expressed by means of *time propositions*, which are just numerals – for ease of exposition Cohen and Levesque represent them as dates. These will be true or false in a course of events at a given index if and only if the index is the same as that denoted by the time proposition. To express that an event ϵ has to happen at time t, they use conjunction, namely, $\epsilon \wedge t$.

Action expressions. They define primitive events, such as physical movements, etc. On top of those they build complex action expressions, by means of the standard operators of dynamic logic [Moore, 1980]: *composition " ;", non-deterministic choice " |", test " ?", repetition " *"*.

Temporal modalities. HAPPENS(a) is true at some index point on a world when there is a subsequent (future) index point on that world such that a describes the sequence of events between the two index points; DONE(a) is true at some index point when there is a previous (past) index point such that a describes the sequence of events between the two index points. HAPPENS(x, a) and DONE(x, a) specify that x is the agent of a. Cohen and Levesque also use $\Diamond$ and $\Box$, glossed as *Eventually* and *Always* respectively.

BELief. They assume the usual Hintikka-style axiom schemata [Halpern and Moses, 1985]:

(1a) $\models \forall x\ BEL(x, p) \wedge BEL(x, (p \rightarrow q)) \rightarrow BEL(x, q)$

(1b) $\models \forall x\ BEL(x, p) \rightarrow BEL(x, BEL(x, p))$

(1c) $\models \forall x \neg\ BEL(x, p) \rightarrow BEL(x, \neg\ BEL(x, p))$

(1d) $\models \forall x\ BEL(x, p) \rightarrow \neg BEL(x, \neg p))$

They introduce KNOW by definition:

Definition 1 $KNOW(x, p) = p \wedge BEL(x, p)$

GOAL. An agent has many, possibly conflicting, desires. Among those he chooses his goals, namely, those desires he wants most to see fulfilled. Goals are modeled as propositions true in all of the agent's chosen worlds. The following proposition holds:

Proposition 1 $\models$ (BEL x p) $\rightarrow$ (GOAL x p)

namely, worlds compatible with an agent's goals must be included in those compatible with his beliefs: given a set of worlds that are possible given what an agent believes, a subset of these are the worlds in which the agent's goals are achieved.

Notice that – I am quoting from [Allen, 1990] – *the formula $GOAL(x,p)$ does not assert that the agent x has p as a goal in the intuitive sense of the word. Rather, it says that p will be true in any world where the agent's goals are achieved. In some ways, the term "consequence-of-goals" might be a better term name for the operator.*

By means of the definition of goal, Cohen and Levesque define the concept of *persistent goal*, and on top of it, the concept of intention.

Persistent goal, *P-GOAL(x, p)*. An agent x has a P-GOAL p if he has a goal p that he believes currently to be false and that he will continue to choose – at least as long as a given condition q remains valid. At first, they define q as saying that the agent won't give up p until he has achieved it, or until he believes that p will never be true. Later, they relax this "fanaticism" requirement by defining a *persistent goal relative to* r, *P-R-GOAL(x, p, r)*, where r is q with a third disjunct u added: the agent will keep the goal p until he has achieved it, or he believes that p will never be true, or he believes that another condition u is not true any more, where u can for example be the reason why the agent adopted p in the first place.

Intention, *INTEND(x, p, q)*. Finally, they define intention as a kind of persistent goal – that is, a persistent goal to do an action, believing one is about to do it, or to achieve some state of affairs, believing one is about to achieve it. Being a *P-R-GOAL*, an intention will be relative to a certain condition q [3].

A model for rational interaction

By means of the logic defined above, Cohen and Levesque develop a model for *rational interaction*. They model agents as being sincere and helpful.

An agent x is *sincere* with respect to another agent y and a certain proposition p, if, whenever x has chosen to do something next in order to cause y to believe p, x has chosen to bring it about that y knows p, i.e. that y believes p, and p holds.

An agent x is *helpful* to another agent y if, for any action α, x adopts y's goal that x will eventually do α. Helpfulness is a disposition only: not taking on another agent's goals does not indicate unhelpfulness, since the agent may have reasons for not wanting the goal. The *helpfulness assumption* is as follows [4]:

Assumption 1
$$\begin{array}{l} \text{HELPFUL(x,y)} \wedge \\ \text{BEL(x, GOAL(y, } \Diamond\text{(DONE(x, } \alpha\text{)))} \wedge \\ \neg\text{(GOAL(x, } \Box\neg\text{(DONE(x, } \alpha\text{))} \\ \qquad\qquad\qquad \longrightarrow \\ \text{INTEND(x, } \alpha, \\ \qquad\qquad \text{[HELPFUL(x, y)} \wedge \text{GOAL(y, } \Diamond\text{DONE(x, } \alpha\text{))])} \end{array}$$

The preceding assumption reads as follows. If agent x

- is HELPFUL to agent y;
- believes that y wants him to perform α, namely, y has the goal that x will eventually perform α;

[3]Actually, they define two operators, INTEND_1 and INTEND_2, distinguished because the object of the intention is respectively actions and states of affairs. Given that I will talk only about intentions to do an action, namely, INTEND_1, I will drop the subscript.

[4]I report formulas from [Cohen and Levesque, 1990b] verbatim, apart from possibly substituting variable names to keep them consistent with the abbreviations I have used in this proposal.

- doesn't have any reason not to perform α, namely, he doesn't have among his own goals that he will never perform α,

then x will commit to α, under the assumption that x remains helpfully disposed and that y doesn't change his mind with respect to x *doing* α.

Cohen and Levesque define the notions of Alternating BELief and Belief in Mutual Belief, ABEL and BMB respectively:

Definition 2 $ABEL(n,\ x,\ y,\ p) = \underbrace{BEL(\ x,\ BEL(\ y,\ BEL\ (x, \cdots\ BEL(\ x,}_{n}\ p\underbrace{)...)}_{n}$

Definition 3 $BMB(\ x,\ y,\ p) = \forall n\ ABEL(\ n,\ x,\ y,\ p)$

Finally, Cohen and Levesque characterize utterance properties. They start by giving a general definition relating an utterance in a certain syntactic mode Φ and the beliefs that H adopts as a consequence of such utterance. Such definition, informally stated, is:

Definition 4 *If H believes that e was just done, where e is the uttering by S of a sentence ϕ in syntactic mode Φ, and if H does not believe that e was done insincerely regarding certain* core attitudes $\mathcal{A}$ *associated with utterances of that type, then H believes that $\mathcal{A}$ hold.*

In the formal definition, ABEL, not BEL, is used, and it is required that ABEL holds at any level of alternating belief. Therefore, if at each level S's sincerity is believed, we obtain that S and H mutually believe $\mathcal{A}$. The properties characterizing the various syntactic modes Φ's are modeled by axioms of the sort

$$\models \Phi(\phi) \Longrightarrow \mathcal{A}$$

As far as imperatives are concerned, Cohen and Levesque attribute them the following property:

> *After speaker S's imperative to addressee H to do action α, if H does not think that S was insincere about his wanting H to do α – that is, if H does not believe that S wanted H to believe falsely that S wants H to do α – then H believes that S wants H to do α.*

This translates to the following domain axiom for imperatives

Axiom 1 $\models$ IMPERATIVE(ϕ) $\Longrightarrow$
GOAL(S, $\Diamond$[$\exists\ \alpha$ [DONE(H, α) $\wedge$
FULFILL-CONDS(ϕ, α)])

namely, S's GOAL when uttering the imperative ϕ is that H performs α which FULFILLS the satisfaction conditions imposed by sentence ϕ. In [1990b, p.235], Cohen and Levesque acknowledge that *FULFILL-CONDS is just a placeholder for a semantic theory that can characterize the meanings of imperatives. The only requirement we make for analyzing imperatives is that such a semantic theory have the capacity to supply predicates (or properties) that are true of events, especially the utterance event itself.*

By applying def. 4 to Axiom 1, we can conclude

$$\text{BMB(S, H, GOAL(S, }\Diamond[\exists\alpha\ [\text{DONE(H, }\alpha) \wedge \text{FULFILL-CONDS(}\phi, \alpha)]]))$$

If we substitute such belief into the second conjunct in the antecedent of Assum. 1, we can conclude:

$$\text{INTEND(x, }\alpha\text{, [HELPFUL(x, y)} \wedge \text{GOAL(y, }\Diamond\text{DONE(x, }\alpha))]$$

where α is the α relative to S's desires.

5.1.2 Extensions to Cohen and Levesque's model

I will now refine the axiom for imperatives given above by providing axioms first for positive imperatives containing purpose clauses and then for negative imperatives. The reader may wonder why the general axiom for imperatives is not sufficient: one might as well leave it as it is, and reason about intentions and objects to intentions in a different part of the model. However, this would not be correct. I claim that, after S utters an imperative containing a purpose clause or a *neg-TC*, H adopts more specific intentions than those predicted by Axiom 1: therefore, the model must be able to capture these more specific effects.

Purpose Clauses

In Ch. 3, I attributed the following property to positive imperatives containing purpose clauses: *Do α to do β* is used to tell H that he should adopt the intention to do β; and that α contributes to achieving β, which in turn constrains the interpretation / execution of α. I also observed that, although S's intention is that H also adopts α as one of his intentions, β has priority: if H adopts another method γ to achieve β, S won't object to it, as long as γ is not against S's intentions and beliefs [5].

Finally, I defined two kinds of accommodation processes, one that computes a more specific action description α', the other that computes the assumptions under which the relation between α and β makes sense.

I will provide two axioms: the former, Axiom 2, models the communicative act between S and H. On the basis of Axiom 2 and other axioms previously established, I will conclude what intentions and beliefs H adopts, upon hearing or reading such an imperative. By applying the other axiom, Axiom 3, to such newly acquired intentions and beliefs, I will model the beliefs and intentions that H holds with respect to the objects of the accommodation inferences.

Axiom 2 is as follows - FULFILL-CONDS-PC is a specialized version of FULFILL-CONDS, $\mathcal{B}$ denotes the purpose clause in ϕ, CONTRIBUTES is a generic name for the relation between α and β that is embodied in purpose clauses; also notice that I am using $\mathcal{A}$ and $\mathcal{B}$ to keep distinct surface form and action expressions:

[5] For example, a teacher telling a student *study more to get better marks*, won't be satisfied if H adopts the "method" of cheating on the exam, in order to get better marks.

Axiom 2 $\models$ Do-$\mathcal{A}$-to-do-$\mathcal{B}(\phi)$

$$
\begin{aligned}
&\Longrightarrow \\
&\text{a) GOAL(S, } \Diamond\ [\exists\ \beta\ [\ \text{DONE(H, } \beta) \wedge \\
&\qquad \text{FULFILL-CONDS-PC(} \mathcal{B}, \beta)]]) \\
&\wedge \\
&\text{b) GOAL(S, } \Diamond\ [\exists\ \alpha\ [\ \text{DONE(H, } \alpha) \wedge \\
&\qquad \text{FULFILL-CONDS-PC(} \mathcal{A}, \alpha) \\
&\qquad \text{BEL(H, CONTRIBUTES (HAPPENS(H, } \alpha), \\
&\qquad\qquad \text{HAPPENS(H, } \beta)))]])
\end{aligned}
$$

Substituting the first conjunct (2.a) into the antecedent of the *helpfulness assumption* – Assum. 1, we get

$$\text{INTEND(H, } \beta, \text{q)}$$

where q are the assumptions of helpfulness etc., as discussed above.

As regards the second conjunct (2.b), it may be questionable whether DONE (H, α) should be included, given my observation above that the goal β has more importance than α: however, I will keep it for the moment. Reasoning exactly as for (2.a), we will obtain

$$\text{INTEND(H, } \alpha, \text{r)}$$

Moreover, from the belief part of 2.b, Def. 4, and an axiom relative to informative acts [6], we will get

$$\text{BMB(S, H, CONTRIBUTES (HAPPENS (H, } \alpha), \text{ HAPPENS (H, } \beta)))$$

I will now introduce the second axiom, that no longer concerns the communicative situation, but characterizes H's mental processes after he has acquired the intentions of doing β and α, and the belief that α contributes to achieving β, from a purpose clause: either H will adopt the intention to perform α', which is α constrained by β; or there will be another belief, ψ, that H has to have in order for the CONTRIBUTES relation to make sense – in the axiom, CONSTRAINED is meant to be true of an action α' which is α constrained by the fact that α CONTRIBUTES to β, and CONTRIBUTES$_1$ is the CONTRIBUTES relation "augmented" with the assumption ψ:

[6] To be defined.

Axiom 3 $\models$ INTEND(H, β, q) $\wedge$ INTEND(H, α) $\wedge$
BEL(H, CONTRIBUTES (HAPPENS (H, α), HAPPENS (H, β)))

$$\Longrightarrow$$

a) [$\exists\alpha'$ [INTEND(H, α', q) $\wedge$
BEL(H, CONSTRAINED
(α', α, CONTRIBUTES (HAPPENS (H, α),
HAPPENS(H, β))) $\wedge$
BEL(H, CONTRIBUTES (HAPPENS (H, α'), HAPPENS (H, β)))]

$\vee$

b) [BEL(H, ψ) $\wedge$
BEL(H, CONTRIBUTES$_1$ (HAPPENS (H, α), HAPPENS (H, β), ψ))]

Disjunct a) is meant to capture the first kind of accommodation, namely, the fact that the action which H has to perform may be a more specialized version of α; notice that INTEND(H, α', q) is supposed to "substitute", in some sense, INTEND(H, α, q). Actually it is consistent that the two intentions coexist, because α is an abstraction of α': however, I wonder whether having them coexist is a perspicuous representation of H's intentions – I will address this point in my future work.

Disjunct b) is meant to capture the second kind of accommodation, the one that takes place when the CONTRIBUTES relation needs an assumption ψ in order to make sense. CONTRIBUTES$_1$ is very similar, at least intuitively, to CGEN – conditional generation – that Pollack defined [1986]: the definition of conditional generation rests on the assumption that there are conditions that have to hold for the generation relation to go through. CONTRIBUTES$_1$ is more general than CGEN, in that it should subsume both generation and enablement, and that there are assumptions that help "make sense" of it; the definitions of CONTRIBUTES and CONTRIBUTES$_1$ are left for future work.

Axiom 3 is clearly descriptive: it just mentions that H believes that a more specific α' contributes to β, or that there is an assumption ψ that under which CONTRIBUTES(HAPPENS (H, α), HAPPENS(H, β)) goes through. However, nothing is said about how α' or ψ are arrived at. Instead of providing inference rules to compute such objects, I'd rather keep the axioms as a way to account for the initial intentions H adopts, and I will use other parts of the model – namely, the inferences that build the plan graph – to compute the actual objects of H's intentions.

Finally, I would like to point out that Axiom 3, as stated, is incorrect: in fact, it may be applied every time H intends a certain β, and he believes there is an α that contributes to β. This would entail, for example, that, if H knows more than one α that contributes to β, H can commit to all such α's, possibly modified to be α'. Instead, this axiom should be applicable only when there is some more evidence for H committing to α', as for example the fact that these intentions and beliefs come from purpose clauses. This applicability condition has to be added.

Negative imperatives

On the basis of the results established in ch. 4, I propose the following characterizations:

DONT imperatives. *Don't do $\mathcal{A}$* is used when S thinks that H is likely to adopt the intention to perform α, and wants to prevent H from adopting it.

Axiom 4 (DONT) $\models$ Don't-do-$\mathcal{A}(\phi)$

$$\Longrightarrow$$
$$\text{GOAL}(S, \Box\, [\neg\, \exists\, \alpha\, [\text{DONE}(H, \alpha) \wedge \text{FULFILL-CONDS-DONT}(\mathcal{A}, \alpha)]])$$

From Axiom 4, the *Helpfulness* assumption, and Def. 4, we can conclude

$$\text{INTEND}(H, \Box\neg(\text{DONE}(H, \alpha)), q)$$

As the reader may recall, there are cases in which *DONT* imperatives behave like *neg-TC* imperatives: in those cases, Axiom 4 should not apply. Given that Axiom 4 describes the default use of *DONT* imperatives, it should probably be restated as an assumption.

Neg-TC imperatives. *"Do $\mathcal{B}$. Take care not to do $\mathcal{A}$"* is used when S wants H to perform $\mathcal{B}$ and thinks that H may NOT adopt the intention of preventing an undesirable consequence of $\mathcal{B}$, or may adopt the intention of performing $\mathcal{B}$ in an undesirable way – where $\mathcal{A}$ describes the undesirable feature to be avoided.

The schema for *neg-TC* imperatives is similar to the one for purpose clauses, therefore highlighting the similarity of these two constructions. I will provide two axioms, Axioms 5 and 6: the former will model the communicative act taking place between S and H, and will allow me to infer the intentions and beliefs that H derives from the utterance. The latter axiom, applied to the conclusions of the former, will model the reasoning that H engages in subsequently.

A first approximation for Axiom 5 follows – AVOID (IN (HAPPENS(H, β), HAPPENS(H, α))) indicates that α is an undesirable feature of β:

Axiom 5 (neg-TC) $\models$ Do-$\mathcal{B}$.-Take-care-not-to-do-$\mathcal{A}(\phi)$

$$\Longrightarrow$$
$$\begin{aligned}&\text{GOAL}(S, \Diamond\, [\exists\beta\, [\, \text{DONE}(H, \beta) \wedge \\ &\qquad\qquad \text{FULFILL-CONDS}(\mathcal{B}, \beta)]]) \wedge \\ &\text{GOAL}(S, \Diamond\, [\exists\alpha\, [\, \text{BMB}(S, H, \\ &\qquad\qquad\qquad \text{AVOID} (\, \text{IN} (\, \text{HAPPENS} (H, \beta), \\ &\qquad\qquad\qquad\qquad \text{HAPPENS} (H, \alpha)))) \wedge \\ &\qquad\qquad \text{FULFILL-CONDS-neg-TC}(\mathcal{A}, \alpha)]])\end{aligned}$$

As for purpose clauses, we want now conclude that, from the intentions and beliefs derived from the utterance of a *neg-TC* imperative, H will derive a further intention to do β modified to avoid α:

Axiom 6 $\text{INTEND}(H, \beta, q) \wedge \text{BEL}(H, \text{AVOID} (\text{IN} (\text{HAPPENS} (H, \beta), \text{HAPPENS}(H, \alpha))))$

$$\Longrightarrow$$
$$\begin{aligned}&\exists\beta'\, [\text{INTEND}(H, \beta', q) \wedge \\ &\qquad \text{CONSTRAINED}(\beta', \beta, \text{AVOID} (\text{IN} (\text{HAPPENS} (H, \beta), \text{HAPPENS} (H, \alpha)))]\end{aligned}$$

As I said, this is just a first attempt to model *neg-TC* imperatives, and there are several problems with it. One is the AVOID operator that I introduced above: it is not clear what it exactly means. One alternative would be to model α as a condition not to be satisfied by β, a condition that could be taken care of by FULFILL-CONDS($\mathcal{B}$, β).

Another problem is that Axiom 5 requires that $\mathcal{B}$ be known. This is the case when the *neg-TC* imperative is contained in an adjunct, so that $\mathcal{B}$ is the action described in the corresponding matrix clause. However, if the *neg-TC* imperative is contained in a matrix clause, it is not known a priori what $\mathcal{B}$ is: H has to understand to which $\mathcal{B}$ $\mathcal{A}$ relates.

5.1.3 Further work on the model

For the time being, I have simply – and very roughly! – sketched axioms for modeling particular kinds of imperatives. Some issues that clearly need to be addressed are:

1. Refine the formalization, making sure that it actually models the imperatives as they should be modeled – for example, I pointed out that the applicability of Axiom 3 should be restricted, that Axiom 4 should probably restated as an assumption, and that there are problems with Axiom 5. I also have to define CONSTRAINED, CONTRIBUTES, CONTRIBUTES$_1$ and AVOID. For example, the operator AVOID may be expressible by means of the already defined action constructors and modal operators; however, this doesn't seem possible as long as there is no way of expressing simultaneity between actions, in the sense of being able to express *While doing β, do (or don't do) α.*
2. The treatment of time is a bit simplistic, as [Allen, 1990] remarks. I have to check what time properties I need in my model: I already mentioned simultaneity.
3. Finally, I need to make sure that H's intentions as inferred by means of the axioms I propose and as depicted in the plan graph are consistent and compatible with each other. For example, the plan graph may include actions that are not explicitly mentioned in the input instructions – what is the status of such actions with respect to the intentions inferred from the axioms?

5.2 The action representation formalism

I will start by making some observations on the features that an action representation formalism should include to be adequate to model NL descriptions of actions – these observations rest on the analysis of purpose clauses and negative imperatives that I presented in the previous chapters, and on analysis of other constructions, such as *free adjuncts* and *check constructions*, on which I reported elsewhere [Webber and Di Eugenio, 1990]. Notice that in the whole discussion *action* is meant as *action type*, and *action description* as *action type description.*

Individual action descriptions.

1. It has probably become obvious by now, but let me point out that, like individuals, sets of individuals, propositions [Bach, 1990] etc. actions should be part of the underlying ontology of the representation formalism. This has also been advocated by Jackendoff, who includes *ACTION* among his ontological categories and justifies it by means of our ability to refer to an action – *I did* **it**, or to ask questions about them – **What** *did you do?* [Jackendoff, 1983].
2. Action descriptions can always be further specified. Consider:

Ex. 46 *a) Apply paste to the wall.*
b) Using a paint roller or brush, apply paste to the wall.
c) Using a paint roller or brush, apply paste to the wall, starting at the ceiling line and pasting down a few feet and covering an area a few inches wider than the width of the fabric.

Therefore the formalism must be able to deal with action description that may not exactly correspond to the stored knowledge, and it must be able to support computation of relations between action descriptions that differ in level of specificity.

3. The formalism must be able to represent not just the usual participants in an action such as agent or patient, but also means, manner, direction, extent etc.

Relations between actions. The formalism must be able to represent various relations between actions, such as temporal relations, *generation* and *enablement*.

Another relation that the formalism should be able to account for is *test*, relating two actions one of which is a test on the outcome or execution of the other. Consider:

Ex. 47 *To attach the wires to the new switch, use the paper clip to move the springtype clip aside and slip the wires into place. Tug gently on each wire* **to make sure it's secure.**

Such constructions are fairly frequent in instructions: they are marked by verbs such as *check, make sure, be certain ...* plus a subordinate clause introduced by the complementizer *that* which describes a state σ. They are used to tell H to check whether a certain state σ holds; pragmatically, they also alert H to the fact that, if σ doesn't hold, he should do something to bring σ about.

In Ex. 47, the purpose clause of the second sentence provides a constraint on the outcome of attaching wires to the new switch – that the attachment be secure. Notice that, consistently with my analysis of purpose clauses, *tug gently on each wire* generates *make sure it's secure*. In this example, the corrective action is not explicit.

In Cohen and Levesque's logic there is an operator for *test*, " ?". However, " ?" is applied to states of the world, not to pairs of actions. Including the *test* relation in my formalism is left for future work.

It seems to me that knowledge about individual actions is typically definitional, while knowledge about relations between actions is not. The problem is how to represent these different kinds of knowledge.

The first possible solution would of course be to adopt a uniform representation system, and therefore disregard such difference; however, this would not take advantage of the different inference mechanisms that are associated with different kinds of knowledge. Consider the following instruction:

Ex. 48 *Cut the square along a perpendicular axis to create two triangles.*

The relation between *cut the square along a perpendicular axis* and *create two triangles* is at least odd, if not ill-formed, because cutting a square along an axis won't create two triangles.

However, the two action descriptions by themselves are perfectly well-formed. In order to track down the ill-formedness above, a homogeneous representation system that does not support distinguishing between different kinds of knowledge could e.g. try to prove that *cut the square along a perpendicular axis* is not a well-formed description, a clearly undesirable behavior.

I will therefore propose to capture definitional information about actions by means of the T-Box of a hybrid system – see next section – and information about relations between actions by means of an *action library*. The semantic primitives of such representation will be the same that Jackendoff defines as part of Conceptual Structures.

In the following, I will first describe hybrid systems, and show that they facilitate computing the first kind of accommodation, namely, inferring a more specific description of an action; then describe Jackendoff's Conceptual Structures, and show how such representation can express knowledge about individual actions, and ultimately to compute the second kind of accommodation, namely, inferring assumptions. Finally, I will show why their integration is necessary.

5.2.1 Hybrid Systems

Hybrid Knowledge Representation systems, such as KRYPTON [Brachman *et al.*, 1983b], KL-TWO [Vilain, 1985], and CLASSIC [Brachman *et al.*, 1990], stemmed from the KL-ONE formalism [Brachman and Schmolze, 1985]. They are composed of two parts: a terminological part, or T-Box, that is used to define terms, and an assertional part, or A-Box, used to assert facts or beliefs.

All T-Box information is definitional in nature. The T-box language has two main categories, *concepts* and *roles*, roughly corresponding to frames and slots. Concepts are organized in a hierarchy of structured terms. The relation between terms in the hierarchy is *subsumption*, which captures the notion that all instances of the subsumed concept are by definition instances of its subsuming concepts.

The T-Box is equipped with a *classification algorithm*, which takes a concept and determines the subsumption relations between it and all the other concepts in a given Knowledge Base. By means of the classification algorithm, a new concept can be automatically assimilated into the taxonomy by "placing it in the right place", i.e. linking it to its most specific subsumers and its most general subsumees. Given that the classification algorithm is able to deal with concepts it has never seen before, the T-Box can be seen as an *infinite virtual* lattice.

The A-box uses first-order logic [7]. The terms of this logic are defined in the T-Box (A-box and T-Box share a symbol table).

To adapt such structures to representing actions, verb phrases have to map to concepts in the T-Box. In the T-box that I will construct, verbs will be concepts decomposed according to Jackendoff's Conceptual Structures.

The A-Box will contain assertions about "individuals", which will be the instances of concepts obtained as a result of the parsing process [8].

[7]The A-Box language is generally somewhat restricted, for example function-free, in order to make theorem proving computationally tractable.

[8]Here assertions should be understood as regarding the content of the NL input, and *individuals* are individual action descriptions, not action tokens – this approach is similar to the one adopted in PSI-KLONE [Brachman *et al.*, 1979].

I will also develop a third box, an *action library*, which will be devoted to representing the rest of the knowledge about actions, such as knowledge of the effects expected to occur when an action of a given type is performed, and information about relations between action-types: temporal, generation, enablement, testing. This knowledge can be seen as *common sense* planning knowledge, which includes facts such as *to loosen a screw, you have to turn it counterclockwise*, but not complex *recipes* to achieve a certain goal [Balkanski, 1990], such as how to assemble a piece of furniture. I would like to stress that the knowledge encoded in the action library relates to *individual action descriptions* and their *relations* to one another, basically to the individual blocks that should be used to build more complex plans; such plans will be contained in the *plan library*, with which I am not currently concerned.

Accommodation: computing more specific actions

A question that needs to be answered at this point is: why use a hybrid KR system, with the added complexity that classification involves, instead of using a simpler representation augmented with an inheritance mechanism? The answer is that in fact I need the power of classification to provide the flexibility to deal with concepts that don't exactly correspond to the stored knowledge.

Let's go back to Ex. 6 – *Cut the square in half to create two triangles.*

Consider a T-Box such as the one shown in Fig. 5.1 – notice that, to keep things simple, I have used intuitively appealing names for concepts and roles: this T-Box is not expressed in Jackendoff's terms, and some of the names I use, such as *location* and *result*, are not linguistically well motivated either.

Given Ex. 6 as input, the individual action description α – *cut (the) square in half* [9] – will be asserted in the A-Box and recognized as an instance of the concept *cut* and as an abstraction of γ – *cut (a) square in half along the diagonal*, as shown in Fig. 5.2. At this point, inference mechanisms that I will describe in more detail later will infer that the action to be performed is actually γ and not α: this can be inferred by exploiting the context in which α appears. The context is composed by β (the goal *create two triangles*), the fact that γ generates β, and the position of α in the T-Box, in particular its position with respect to γ. This implements one of the two kinds of accommodation that I described in ch. 2 – see fig. 2.2.

Notice that by exploiting the classification process we can also deal with cases in which α is more specific than γ, or in which α is in conflict with γ, such as in Ex. 48. In this case we can recognize that γ = *cut the square in half along the diagonal* and α = *cut the square along a perpendicular axis* are in conflict, because the role fillers of *location* on α and γ are non-unifiable, being *along(perpendicular-axis)* and *along(diagonal)* respectively. I am not planning to address the issue of what strategies should be adopted in case such a conflict is detected: presumably, some kind of interaction with the user could be triggered.

In conclusion, given *Do* α *to do* β, and a stored generation relation GEN(γ, β, *agent*, *t*) – see sect. 5.2.3 on the representation of *generation* and *enablement*:

1. If γ is an ancestor of α, α is the action to be performed.
2. If α is an ancestor of γ, we can assume that γ is the action to be performed.

[9] As I already mentioned, I assume there is a discourse model that takes care of definite reference etc.

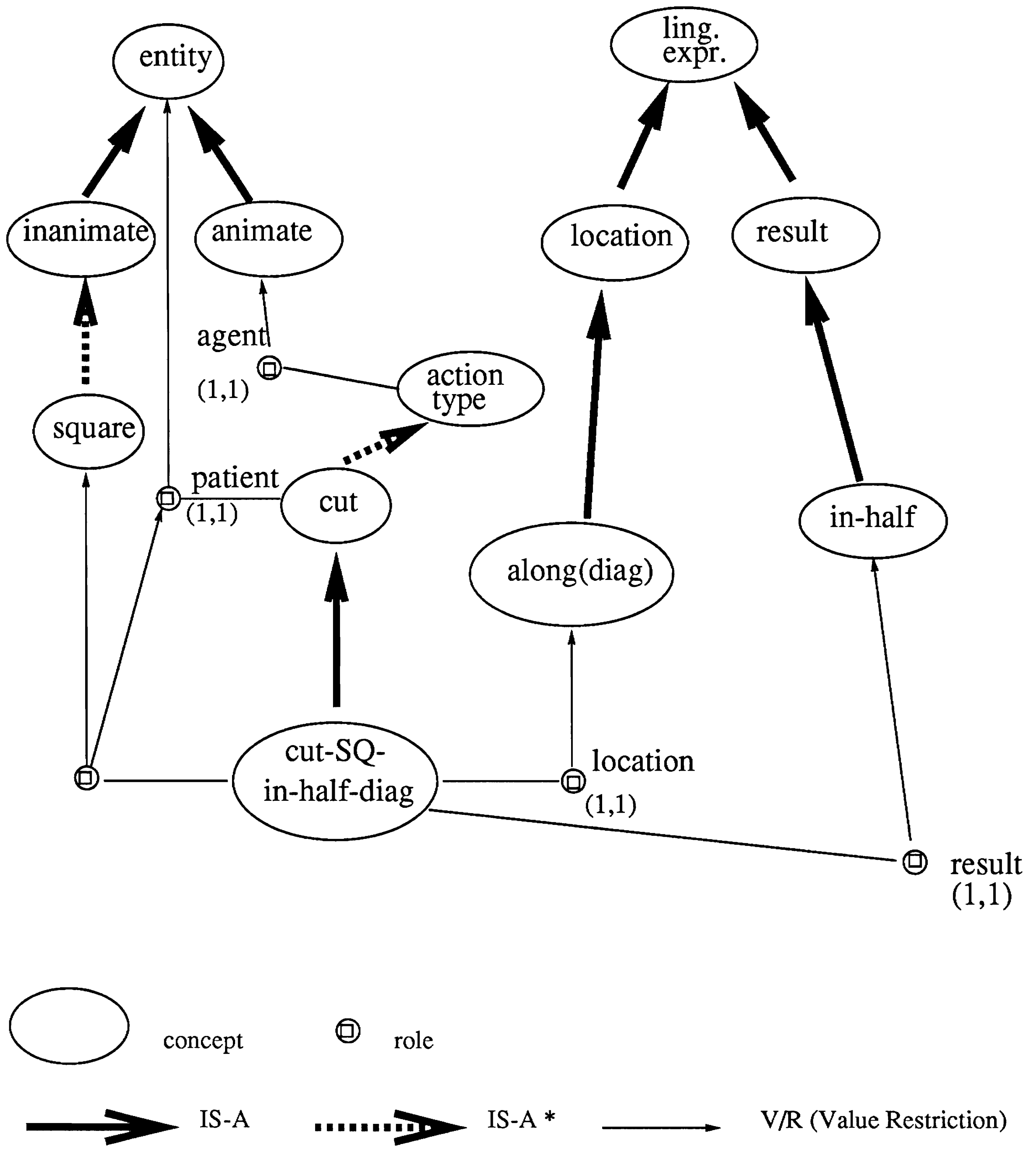

Figure 5.1: A portion of the action hierarchy

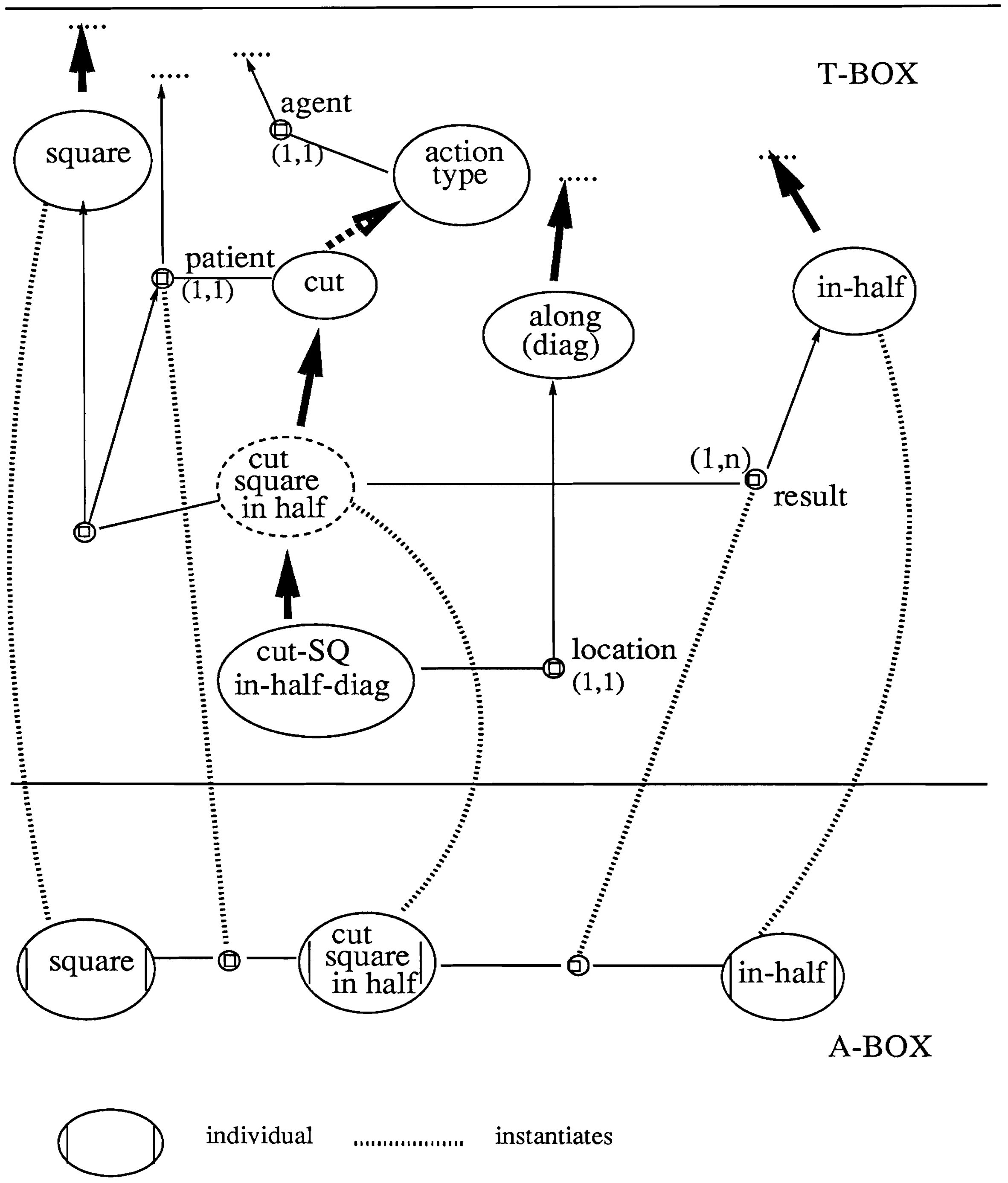

Figure 5.2: Dealing with less specific action descriptions

3. If α and γ are not ancestors of each other, but they can be unified – i.e. all the information they provide is compatible, as in the case of *cut the square carefully* and *cut the square in half along the diagonal* – then their unification $\alpha \sqcup \gamma$ is the action to be executed.

4. If α and γ are not ancestors of each other, and they provide conflicting information – such as *cut the square along the axis* and *cut the square along the diagonal* – then signal *failure.* This could trigger interaction with the user.

5.2.2 Jackendoff's Conceptual Structures

Work done in lexical semantics, in particular Jackendoff's Conceptual Structures [Jackendoff, 1990], proves to be very useful to represent action descriptions in a linguistically motivated way, as [Di Eugenio and White, 1991] shows. As we will see, there is significant mileage to be gained from using a decompositional theory of meaning, because the semantic primitives do manage to capture important generalizations. In the rest of this section, I will introduce the notation and some minor modifications to the theory as presented in [White, 1991], and I will apply the notation to the representation of the clause *Go into the other room* [10].

In Jackendoff's theory, an entity may be of ontological type *Thing, Place, Path, Event, State, Manner* or *Property*. The conceptual structure for a room is shown in (5.2a) below:

(2a) $[_{\text{Thing}}\ \text{ROOM}]$

(2b) $[_{\text{Thing}}\ \text{KITCHEN}]$

Square brackets indicate an entity of type *Thing* meeting the enclosed featural description. Small caps indicate atoms in conceptual structure, which serve as links to other systems of representation; for example, the conceptual structure for a kitchen (5.2b) differs from that of a room only in its choice of constant. Jackendoff leaves the determination of their similarities and differences to a system of representation better suited to the task, able to address perceptual distinctions.

To distinguish instances of a type, in [Zwarts and Verkuyl, 1991] it is required that every conceptual structure have an index, as in (5.3):

(3) $[_{\text{Thing}}\ \text{ROOM}]_1$

Conceptual structures may also contain complex features generated by conceptual functions over other conceptual structures. For example, the conceptual function IN: Thing $\rightarrow$ Place may be used to represent the location *in the room* as shown in (5.4a) below. Likewise, the function TO: Place $\rightarrow$ Path describes a path that ends in the specified place, as shown in (5.4b):

(4a) $[_{\text{Place}}\ \text{IN}([_{\text{Thing}}\ \text{ROOM}]_k)]_l$

(4b) $[_{\text{Path}}\ \text{TO}([_{\text{Place}}\ \text{IN}([_{\text{Thing}}\ \text{ROOM}]_k)]_l)]_m$

(4c) $[_{\text{Path}}\ \text{TO}(l)]_m$

(5.4c) is an equivalent representation of (5.4b), where the index l stands for the entire constituent. This move considerably lessens the typographical burden of representing large con-

[10]All modifications to Jackendoff's theory have been developed by Mike White.

ceptual structures; to further lessen this burden, indices and ontological types will often be left out.

To complete our clause[11], it remains only to add the conceptual function GO: Thing × Path → Event:

(5) $[_{\text{Event}}\ \text{GO}([_{\text{Thing}}\ ]_i,\ m)]$
$[_{\text{Path}}\ \text{TO}([\text{IN}([\textsc{other-room}])])]_m$

As there is no explicit subject in our clause, the constituent i (pragmatically, the AGENT) in (5.5) is left unspecified.

To distinguish *Walk into the other room* from (5.5), an indication of manner should be included:[12]

(6) $\left[\begin{array}{l}\text{GO}(i, m)\\ [_{\text{Manner}}\ \textsc{walking}]\end{array}\right]$

The final modification to Jackendoff's theory is the addition of a new semantic field. Semantic fields, such as Spatial and Possessional, are intended to capture the similarities between sentences like *Jack went into the other room* and *The gift went to Bill*, as shown in (5.7) below:

(7a) $[\text{GO}_{\text{Sp}}([\textsc{jack}], [\text{TO}([\text{IN}([\textsc{other-room}])])])]$

(7b) $[\text{GO}_{\text{Poss}}([\textsc{gift}], [\text{TO}([\text{AT}([\textsc{bill}])])])]$

The idea is that verbs like *go* leave the semantic field underspecified, whereas verbs like *donate* specify a particular field. In addition to these semantic fields, White proposes to add a new one called Control. It is intended to represent the functional notion of *having control over* some object. For example, in sports, the meanings of *having the ball*, *keeping the ball*, *getting the ball*, and *losing the ball* embody this notion, and are clearly quite distinct from their Spatial and Possessional counterparts. The inclusion of this new field let's us capture these additional similarities in an analogous way. The similarity between *Jack has the money* and *Jack has the ball*, for instance, is shown in (5.8):

(8a) $[\text{BE}_{\text{Poss}}([\textsc{money}], [\text{AT}([\textsc{jack}])])]$

(8b) $[\text{BE}_{\text{Ctrl}}([\textsc{ball}], [\text{AT}([\textsc{jack}])])]$

5.2.3 The action library

The action library contains both simple plans that represent common sense knowledge about individual actions, and relations such as generation and enablement between individual actions. These plans are used as building blocks that can be manipulated to form more complex plans, that will be contained in a separate plan library.

[11]Ignoring, of course, the meaning of *other* for now.

[12]Though this is clearly intended, Jackendoff never explicitly represents such a distinction.

Header
$[\mathrm{CAUSE}([\textsc{agent}]_i, [\mathrm{GO}_{\mathrm{Sp}}(j, k)])]$ $\begin{bmatrix} \mathrm{FROM}([\mathrm{AT}(j)]) \\ \mathrm{TO}(l) \end{bmatrix}_k$
Body
- $[\mathrm{GO}_{\mathrm{Sp}}([i, [\mathrm{TO}([\mathrm{AT}(j)])]])]_{\gamma_1}$ - $[\mathrm{CAUSE}(i, [\mathrm{GO}_{\mathrm{Ctrl}}(j, [\mathrm{TO}([\mathrm{AT}(i)])])])]_{\gamma_2}$ - $\begin{bmatrix} \mathrm{GO}_{\mathrm{Sp}}(i, k) \\ [\mathrm{WITH}(j)] \end{bmatrix}_{\gamma_3}$ **- Annotations -**   - γ_1 *enables* γ_2 *enables* γ_3
Qualifiers
- $[\mathrm{NOT\ BE}_{\mathrm{Sp}}(j, l)]$
Effects
- $[\mathrm{BE}_{\mathrm{Sp}}(j, l)]$

Figure 5.3: A *Move Something Somewhere* Action.

Individual actions

I will refer to the *move*-action library entry shown in Figure 5.3, which might be described as follows: go to where j is, get control over it, then take it to l – from [Di Eugenio and White, 1991].[13]

Actions have a *header* and a *body*, both expressed in terms of Jackendoff's semantic primitives. The terminology, *header* and *body*, is reminiscent of STRIPS [Fikes and Nilsson, 1971]; however I express the relations between these components in terms of *enablement* and *generation*—for example the body *generates* its header.

The representation does not employ preconditions, because preconditions concern states, but instructions describe actions far more frequently than states. More importantly, it is very

[13]This do-it-yourself method is but one way to move something from where it is to somewhere else. Other methods would be listed separately in the action library.

difficult to draw the line between what is a precondition and what is part of the body of the action. One could say that the body of a *move*-action simply consists of a *transfer* of an object from one place to another; and that a precondition for a *move*-action is having control over that object. However, consider a heavy object: the agent will start exerting force to lift it, and then carry it to the other location. It is not obvious whether the lifting action is still part of achieving the precondition, or already part of the body. Therefore, my point of view is that there are only actions, which may be substeps in executing another action – namely, they may belong to a sequence that generates the header [14]. Actually, the name *sequence* may be misleading, because it conveys the idea of a total temporal order holding between the actions belonging to the sequence. Actions in a body may have other relations holding between them, such as enablement in Fig. 5.3, and the order may be partial. The *annotations* on the body specify the relations between the subactions.

From the planning tradition, I retain the notions of *qualifiers* and *effects*. Qualifiers are conditions that make an action relevant: for example, *unplug x* is relevant only if x is plugged. Qualifiers are useful for computing assumptions—see sec. 5.2.3.

Notice the importance of using a representation such as Jackendoff's: it helps us capture the common characteristics of different actions, e.g. *get* and *carry*. The semantic representation for *carry* would also match the generic *move*-action template, and would add to it a qualification such as

(9) $[_{\text{Manner}}\ \text{WITH}([_{\text{Thing}}\ \text{HANDS}])]$

Having such a representation is also useful for computing qualifiers and effects in a systematic way: they can be precompiled from the representation itself. For example, for every action including a component δ of j moving from where it is to l in its header, i.e.

$$\left[\text{GO}_{\text{Sp}}\left(j, \begin{bmatrix} \text{FROM}([\text{AT}(j)]) \\ \text{TO}(l) \end{bmatrix}\right)\right]_{\delta}$$

we know that after δ, j must be at l, therefore we can include this in the effects of the action. Given the further restriction that j cannot be in two places at once, we may infer that j cannot be at l now, and thus precompute the qualifier.[15]

Relations between actions

We have already seen that relations such as *generation*, *enablement* and *temporal relations* are used in the body of an action. Such relations, in particular generation, are also used to express simple common sense knowledge about actions, such as the simple plan *to loosen a screw, turn it counterclockwise* [16]. I will now discuss these relations in more detail.

- *Temporal relations*: I will adopt relations derived from Allen's temporal logic [1984]. Notice that the formalism must be able to represent possibly underspecified temporal relations. Consider:

[14]On preconditions, see also Pollack's position [1986], which I discussed in ch. 3.

[15]Jackendoff suggests something analogous with his inference rules, which have yet to be formalized.

[16]The actions appearing in these simple plans must be defined in other parts of the action library.

Ex. 49 *Pour mixture over cheese in casserole, spreading evenly.*

Although this description can be seen as a single action (*pouring* having *spreading* as a side-effect) or as two separate actions, consider it from the latter point of view. It is clear that *spreading* has to begin after *pouring* has begun, but the temporal relationship of the two actions is not otherwise constrained. An agent could pour the mixture and, after having poured it, spread it; or hold the receptacle that contains the mixture with one hand, pouring it and simultaneously spreading the mixture with the other hand; or pour the mixture and simultaneously have a second agent spread it. The choice, possibly constrained by the state of the world, will be made at the moment of executing the action.

Such underspecificity can be represented in Allen's logic presumably by means of disjunction, e.g.

$$\mathrm{BEFORE}(t_{pour}, t_{spread}) \vee \mathrm{OVERLAP}(t_{pour}, t_{spread})$$

where BEFORE(t_1, t_2) means that t_1 is before t_2, and they don't overlap in any way; OVERLAP(t_1, t_2) means that t_1 starts before t_2, and they overlap.

- *Generation.* Recall that the defining conditions for generation are: α and β are simultaneous, α is not part of doing β, and when α occurs, a set of conditions C hold, such that the joint occurrence of α and C imply the occurrence of β.

  As a starting point, I adopt the representation of generation put forward by Pollack – I will modify it if the need arises:

Definition 5 $\mathrm{GEN}(\alpha, \beta, \mathrm{agent}, \mathrm{t}) \Longrightarrow$

$$\begin{aligned} \exists\, \mathrm{C}\,[\quad & \text{(i)}\ \forall\, agent_1, \forall t_1\ [\ \mathrm{HOLDS}(\mathrm{C}, t_1) \wedge \mathrm{OCCURS}(\alpha, agent_1, t_1) \\ & \qquad\qquad \rightarrow \mathrm{OCCURS}(\beta, agent_1, t_1)] \\ & \text{(ii)}\ \exists\, agent_2, \exists t_2\ [\mathrm{OCCURS}(\alpha, agent_2, t_2) \wedge \\ & \qquad\qquad \neg\, \mathrm{OCCURS}(\beta, agent_2, t_2)] \\ & \text{(iii)}\ \exists\, agent_3, \exists t_3\ [\mathrm{HOLDS}(\mathrm{C}, t_3) \wedge \neg\, \mathrm{OCCURS}(\beta, agent_3, t_3)] \wedge \\ & \text{(iv)}\ \mathrm{HOLDS}(\mathrm{C,t})] \end{aligned}$$

The four clauses in Def. 5 read as follows:

(i) if condition C holds and action α occurs at time t_1, then action β occurs at the same time t_1;

(ii) α by itself doesn't entail β, namely, α can occur at time t_2 without β occurring at time t_2;

(iii) condition C doesn't entail β by itself, namely C may hold at time t_3 without β occurring at time t_3;

(iv) C holds at time t.

Nobody has ever spelled out how to decide what conditions affect a given generation relation. Such conditions are generally taken to include the fact that an agent is in *standard conditions* with respect to a given action – for example, an agent whose feet are tied won't be able to *go into the other room*; and relevant conditions about the state of the world, for example that, in order for *flipping the switch* to generate *turning the light on*, all electrical equipment must be working. I just want to point out that characteristics of the actions taking part in a given generation relation should not be part of such conditions. It could be in fact argued that some of these characteristics, e.g. *along(diagonal)* for *cutting the square*, should be part of conditions on the *generation* relation. However, it would then be impossible to choose in a principled way which features of an action belong to the description of the action, and which belong to the conditions on generation. Moreover, there are cases in which such features are implicit arguments of verbs. *Turn* has an intrinsic argument, *direction*: therefore, *counterclockwise* has to be a specification of *turn screw*, not a condition on the *generation* relation between *turn screw* and *loosen screw*.

Another observation on conditions: the assumptions I mentioned as an object that accommodation has to compute may again be seen as part of conditions on a given generation relation. However, they seem to me to be of a different nature. In the *Go into the other room to get the urn of coffee* example [17] *go into the other room* may successfully contribute to *getting the urn of coffee* even if there is no urn in the other room – if for example in the other room there is a note saying how to get the urn of coffee. The assumption *urn in the other room* is simply an expectation that H comes to have as a consequence of the accommodation process.

- *Enablement.* In ch. 3, I mentioned the following characteristics of enablement:

  1. If α enables β, α brings about a set of conditions that are necessary, but not sufficient, for the subsequent execution of β.
  2. As a consequence, enablement embodies a notion of temporal precedence: if α enables β, α has to temporally precede β, namely, α has to begin, but not necessarily end, before β: in the following example, *hold* has to continue for the whole duration of *fill.*

     Ex. 50 *Hold the cup under the spigot to fill it with coffee.*
  3. In the same way that the generatee affects the execution of the generator, so the enabled action affects the execution of the enabling action. *to* in *go to the mirror* is interpreted differently, depending upon whether the action to be enabled is *seeing oneself* or *carrying the mirror somewhere else.*

  To my knowledge, the only definition of enablement existing in the literature is Balkanski's definition of conditional enablement. Balkanski's definition reflects her observation that [1990, p.26]

  > *A closer look at the enabling relation shows that the set of conditions brought about by the occurrence of the enabling activity is necessary either to satisfy an executability constraint on the enabled activity or provide the ... condition on a generation relation in which the enabled activity participates.*

  Balkanski exemplifies the former type of enablement with *buying ingredients enables preparing the dish*, and the latter with *inserting the dowels enables attaching the rails.* Notice in fact that *inserting dowels* brings about some conditions under which a third action, such as *hammering*, generates *attaching the rails.* The formal definition is as follows:

[17] This is an indirect generation relation; however, I think the point I am making holds for direct generation as well.

Definition 6 CENABLES(α, β, C_1) $\Longleftrightarrow \exists$ C_2
(i) CGEN(α, ACHIEVE(C_2), C_1) $\wedge$
(ii)[CEXEC(β, C_2) $\vee$ $\exists\gamma$ CGEN(γ, β, C_2)]]

α is defined as enabling β if

1. α generates achieving C_2, and
2. (i) either C_2 has to hold for β to be executable – CEXEC(β, C_2). This is the case for α = *buying ingredients* and β = *preparing dish*;
   (ii) or there is a third action γ that generates β under C_2. This is the case for α = *inserting dowels*, β = *attach rails*, γ = *hammer rails.*

The notion of temporal precedence between α and β that enablement embodies is missing in Def. 6, and is contained in a second definition – ENABLES – that builds on CENABLES and that relates two activities. In fact, both Pollack and Balkanski assume two different types for actions. The first is *act-types*, which don't have an associated performance time, the second *activities*, defined as act-types plus an agent and a time interval. I have doubts on the previous definitions of enablement and generation because I am not convinced that distinguishing between act-types and activities is correct: for example, I don't agree with giving more prominence to the agent role over the other arguments of the verb, as the definition of *activity* seems to advocate.

However, as in the case of generation, I will start from this definition of enablement and I will modify it as the need arises.

As I also mentioned in Ch. 3, the distinction between generation and enablement becomes blurred when we extend the concept of generation to a sequence of actions $\mathcal{A} < \alpha_1, \alpha_2, ..., \alpha_n >$ that generate another action β. Is the relation between $\alpha_i \in \mathcal{A}$ and β generation or enablement? Consider for example the *move*-action in fig. 5.3: what is the relation between the header - interpreted as *get the urn of coffee* – and the actions which are part of its body, for example GO_{sp}, namely, *go into the other room*? The answer seems to stem from intuitions: if one sees the two actions as separate, one could say that *go into the other room* enables *get the urn of coffee*; otherwise, that *go into the other room* is part of *get the urn of coffee.* The question seems even more difficult to answer if, instead of considering α_1 and β, one takes α_i with $i > 1$: it seems very hard to say that α_i enables β.

I think that the only sensible way of answering such question is to do it with respect to the formal representation: I will keep the term *generation* for a relation defined between two actions, or between a sequence $\mathcal{A} < \alpha_1, \alpha_2, ..., \alpha_n >$ and an action β. In the latter case, the relation between α_i and β will be termed *indirect generation* – it will be called *substep* in the plan graph. Enablement will instead hold of pairs of actions α_i and α_k, possibly belonging to a sequence $\mathcal{A}$ generating β.

Accommodation: Making an Assumption

In this section, I will show how the process of understanding Ex. 2 – *Go into the other room to get the urn of coffee*, which requires the assumption that the urn is in the other room, is carried out.

The process begins with the following representation [18]:

$$\begin{bmatrix} \text{GO}_{\text{Sp}}([\textsc{agent}]_i, [\text{TO}([\text{IN}([\textsc{other-room}])])]) \\ \text{FOR}(\beta) \end{bmatrix}_{\alpha}$$

$$[\text{CAUSE}(i, [\text{GO}_{\text{Sp}}([\textsc{urn-of-coffee}]_j, k)])]_{\beta}$$

$$\begin{bmatrix} \text{FROM}([\text{AT}(j)]) \\ \text{TO}(l) \end{bmatrix}_{k}$$

Here the FOR-function (derived from the *to*-phrase) encodes the CONTRIBUTES relation holding between the *go*-action α and the *get*-action β.

Given the presence of the *to* phrase, we know that the *go*-action α may generate or enable the *get*-action β. Given that no direct generation or enablement relation between the two is found, the hypothesis that α may be part of a sequence of actions that generate β is put forward. This hypothesis is pursued first of all by looking up the *get*-action in the action library: β matches the header of the general *move*-action shown in Figure 5.3 if the object j to be moved is bound to *the urn of coffee* [19]:

$$j \quad = \quad [\textsc{urn-of-coffee}]$$

The next step is to try to match the *go*-action with some subaction γ of the *get*-action: the *go*-action can be understood to be the first action γ_1 in the *get*-action by taking $[\text{AT}(j)]$ and $[\text{IN}([\textsc{other-room}])]$ to be the same place. This is equivalent to making the following assumption:

(10) $[\text{BE}_{\text{Sp}}(j, [\text{IN}([\textsc{other-room}])])]$

Assumption (5.10) could of course be wrong, say if there were a note in the next room saying *ha ha, it's not really in this room but the next.*

5.2.4 Integrating Hybrid Systems and Conceptual Structures

So far, I have shown that two different representation schema are useful to perform different kinds of inferences. Moreover, each responds to some of the desiderata I listed earlier: in particular, a hybrid system provides the flexibility required in dealing with action descriptions that don't exactly match the stored knowledge; a semantic representation such as Jackendoff's is linguistically motivated and manages to capture generalizations, such as that *carry* is a *move*-action augmented with a specific physical means of moving the object.

Given that the two formalisms both show promise, the next natural step is to integrate the two. Both will benefit from this integration.

[18]This representation is constructed by a Combinatory Categorial Grammar parser – see [White, 1991] and sect. 5.5.

[19]Notice that the description *the urn of coffee* actually refers to a discourse referent: I assume there is some process responsible for the discourse model, as in fact happens in the AnimNL system – see sect. 5.5.

Defining terms in the T-Box by means of linguistically sound primitives will transform the T-Box into a real lexicon.

On the other hand, a KL-ONE style representation will make it possible to use Conceptual Structures in a computational framework, by endowing it with a hierarchical organization and with the possibility of extending the lexicon. A flavor of hierarchical organization is present in [Jackendoff, 1990] itself. Consider (5.2), repeated here for convenience:

(11a) $[_{\text{Thing}}\ \text{ROOM}]$

(11b) $[_{\text{Thing}}\ \text{KITCHEN}]$

ROOM and KITCHEN are atoms in Conceptual Structures, and so are many other entities, such as LIQUID and WINE. It could be argued that this is too high a level of atomicity: Jackendoff postulates a type *Thing* which is a collection of atoms, including, for example, LIQUID and WINE. Although he mentions that there are rules to determine that WINE satisfies the feature LIQUID, he never goes into the algorithm for performing such computation. This can of course be done by means of a taxonomy rooted in the concept *thing*. *Liquid* will be defined as a subconcept of *thing*, and *wine* as a subconcept of *liquid*, e.g. by adding to *liquid* the restriction *made from grapes*.

By having such a taxonomy, we can also define a concept *animate*, and use it to indicate that the structural position corresponding to *agent* is not *thing* but *animate*, as many theories of action require.

Jackendoff defines all these concepts as atoms because they cannot be defined at the level of Conceptual Structures: this is readily captured in KL-ONE by defining them as primitive concepts, namely, concepts for which necessary but not sufficient conditions can be stipulated – apart maybe from *wine*, as *made from grapes* can be considered as a defining condition.

A hierarchical organization is also very helpful to deal with actions that differ in MANNER, such as *move* and *carry*, which adds to *move* the specification $[_{\text{Manner}}\ \text{WITH}([_{\text{Thing}}\ \text{HANDS}])]$, or go_{sp} and *walk*.

In fig. 5.4, I present part of a T-Box that includes the *move-action* shown in Fig. 5.3. There are four taxonomies, rooted respectively in *thing, place, path, event* [20].

Thing. The taxonomy rooted in *thing* is very straightforward; here only the subconcepts *animate* and *inanimate* are shown.

Place. The concepts belonging to this hierarchy correspond to conceptual functions of the form F: Thing $\rightarrow$ Place. Some such functions are AT, IN, ON. In Fig. 5.4, I show only the concept *at-place*, corresponding to the AT conceptual function. *at-place* has a single role *at* with exactly one filler, of type *Thing*.

Path. The concepts belonging to this hierarchy represent functions yielding *Paths*. The *from-to-path* concept has two roles, *from* and *to*, each of which has a filler *place*. *from-to-path* corresponds to the complex Conceptual Structure:

$$\begin{bmatrix} \text{FROM}([\text{PLACE}]) \\ \text{TO}([\text{PLACE}]) \end{bmatrix}$$

[20] Jackendoff also defines *state, manner, property*. The respective hierarchies are not shown in fig. 5.4.

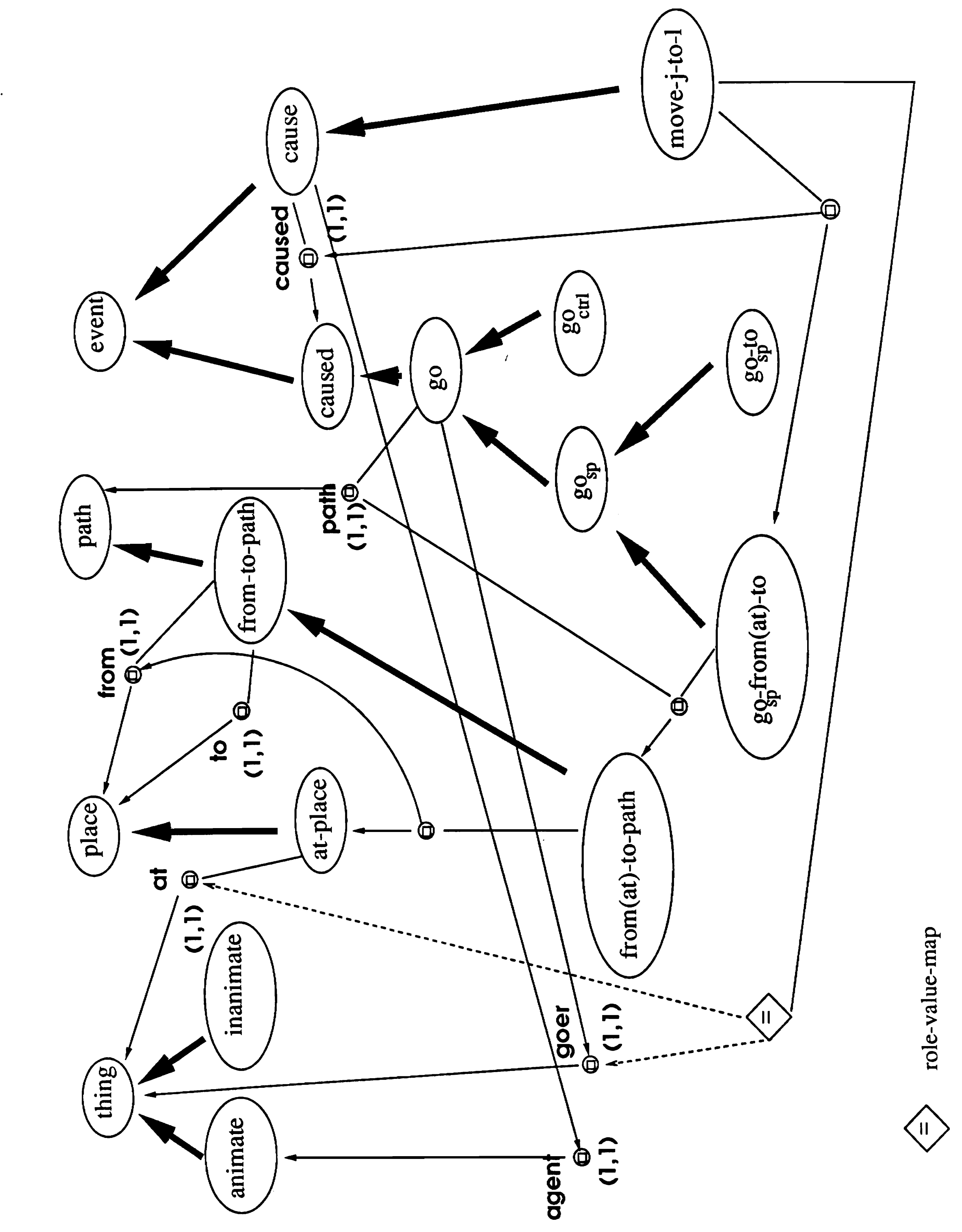

Figure 5.4: A T-Box expressed in terms of Conceptual Structures

The concept *from(at)-to-path* restricts the role *to* inherited from *from-to-path* to be filled by *at-place*. It therefore corresponds to

$$\begin{bmatrix} \text{FROM([AT([}\textsc{place}\text{])])} \\ \text{TO([}\textsc{place}\text{])} \end{bmatrix}$$

Event. Subtypes of *event* are *cause* and *caused*.

cause has two roles, the *agent*, restricted to be *animate*, and the *caused* event.

The hierarchy rooted in *caused* will include all possible events that can be arguments to the conceptual function *CAUSE*. Here I have shown part of the subhierarchy rooted in *go*, which corresponds to the following conceptual function: GO: Thing $\times$ Path $\rightarrow$ Event. The concept *go* has two roles: *goer*, which has *thing* as filler, and *path*, that has a *path* as filler.

Subconcepts of *go* are go_{sp} and go_{ctrl}. The concept go_{sp}*-from(at)-to* restricts the role *path* of *go* to be a *from(at)-to-path*.

Finally, the concept *move-j-to-l* is defined as a subconcept of *cause* by imposing the restriction that the filler of *caused* be go_{sp}*-from(at)-to*. A final restriction is needed, indicated by the role value map: the filler for the role found at the end of one of the branches has to be the same found at the end of the other branch [21].

Notice that this definition of *move-j-to-l* exactly captures its Conceptual Structure definition, as shown in the header in Fig. 5.3, and repeated here for convenience:

$$[\text{CAUSE}([\textsc{agent}]_i, [\text{GO}_{\text{Sp}}(j,k)])]$$
$$\begin{bmatrix} \text{FROM}([\text{AT}(j)]) \\ \text{TO}(l) \end{bmatrix}_k$$

I hope I have convinced the reader that mapping Conceptual Structures into KL-ONE like primitives is rather straightforward, although the final result may not correspond to the intuitions we have regarding those concepts - on the other hand, a Conceptual Structure representation of a verb is not very intuitive either!

The important question now is: with a classical predicate argument representation, such as *move(i, j, TO(l))*, it is very easy to classify new concepts. However, now the representation is in a sense "multi-level": for example, l, the place where to move j, is found navigating from the concept *move-j-to-l* through the role chain *caused, path, at*. I have to verify whether the classifier is able to properly classify these "multi-level" concepts.

5.3 The plan graph

Finally, I want to discuss the third component of my model of instruction understanding, the *plan graph* and the inferences that build it.

[21] In the CLASSIC system that I have been using, role value maps can be expressed by means of SAME-AS, in this case SAME-AS (caused goer) (caused path at). It should be read as *the filler of the role "goer" of "caused" is the SAME-AS the filler of the role "at" of "path" of "caused".*

The plan graph represents the structure of the intentions that the agent adopts as a response to the instructions. It keeps track of the goals the agent is pursuing, of the hierarchical relations between the goals and the actions the agent is going to execute to achieve such goals, and of various relations between actions. It also helps in interpreting the instructions that follow. In the following example, establishing the initial goal *get the urn of coffee* provides the context in which the other instructions have to be interpreted:

Ex. 51 *a) Go into the other room to get the urn of coffee.*
b) Before you pick it up, be sure that it's unplugged.
c) When you bring it back here, carry it with both hands.

A similar strategy is adopted for example by Kautz [1990]: however, he only uses two relations between events, *is-a* and *part-of*, while my representation is richer. In Figure 5.5, the plan graph built after interpreting the three instructions belonging to Ex. 51 is shown: in sect. 5.2.3, I have discussed how the leftmost branch of the tree is built. The algorithms that build the other two branches still need to be designed in detail.

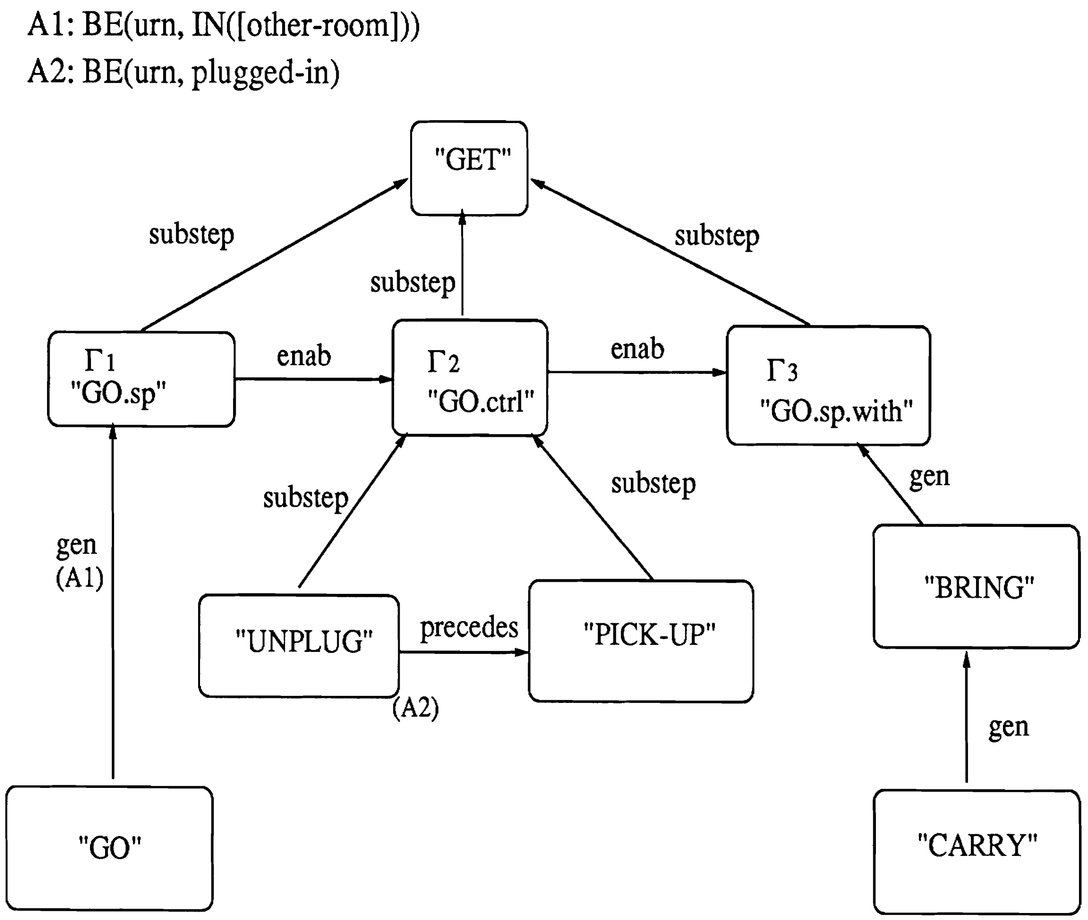

Figure 5.5: The plan graph.

5.3.1 The data structure

The plan graph is composed of nodes that contain descriptions of individual actions, and arcs that denote relations between these actions.

A node contains the Conceptual Structure representation of an action – not the whole entry in the action library, only the header – augmented with the consequent state achieved after the execution of that action; the consequent state is computed from the effects of the action, and is needed for the interpretation of the instructions that follow. Notice that in Figure 5.5 the labels on the nodes are only mnemonics, and do not represent their real contents.

The arcs represent various relations between actions. If α and β are linked by R, with α the origin of R, R can be:

1. *Temporal*, such as *precedes* in Fig. 5.5. Another temporal relation of interest is *during*, that indicates simultaneity. Also, in future extensions to the plan graph in which states are represented, an arc *time* will be used to link an action and a state: its meaning is that the action has to start when a given state comes to hold – this would model instructions such as *turn the gas off when the tofu has turned golden.*

2. *Generation* and *substep*. *Substep* means that α belongs to a sequence that generates β. If the sequence is composed by a single action, *substep* reduces to *generation.*

3. *Enablement*.

4. *And, or*. These arcs may be needed to represent actions belonging to a certain set: *and* will link two actions that both have to be performed, but that are totally unrelated to one another, such as *Prepare a dessert and do laundry* [22]; two actions are linked by an *or* arc if they are in alternative, such as *vacuum or dust-mop the parquet*, from Ex. 23.

There are assumptions associated with the plan graph. Some of those are derived from the accommodation process; in this case, an assumption is associated to a relation between two actions, and therefore, to the corresponding arc in the plan graph—for example A1 in Fig. 5.5. Other assumptions are derived from the qualifiers associated with actions, and are associated with the nodes describing those actions—A2 in Fig. 5.5.

5.3.2 The algorithm

There are various inference processes that manipulate the plan graph, and that can be characterized as either *planning*—e.g. plan expansion, subgoaling—and *plan recognition*—inferring the more abstract goal some actions are supposed to achieve, performing accommodation inferences. Some of these inferences can be made independently of the situation the agent finds him/herself in, others instead require grounding the instructions in the current situation. For example, upon observing that the door to the *other room* is closed, the agent will infer s/he needs to perform *open the door.* As I already said, the only inferences I am interested in are performed prior to grounding the plan in the current situation. Given that the plan graph is used by the AnimNL

[22]The fact that they are unrelated may hold only at the understanding level; when execution time comes, the agent will possibly have to make choices, for example which action to execute first, or even which subactions to interleave.

```
1) Add to A-Box individual(s) corresponding to Conceptual Structure
representation of action(s) described in input instructions.
Set flag ACCOM if they don't exactly match known concepts.

2) IF new utterance is simple imperative
      THEN simple-imperative-procedure;
   IF new utterance is Do-$\alpha$-to-do-$\beta$
      THEN Do-$\alpha$-to-do-$\beta$ procedure;
   IF new utterance is DONT-imperative
      THEN DONT-procedure;
   IF new utterance is neg-TC-imperative
      THEN negTC-procedure;
   IF new utterance contains temporal subordinate
      THEN TEMP-procedure;
   IF new utterance is ...
      THEN ....
```

Figure 5.6: The top-level algorithm

system, other modules expand the plan graph derived from the input instructions according to the current situation – see sect. 5.5.

The plan graph is built by an interpretation algorithm that works by keeping track of *active nodes* – which will include the goal currently in focus, the nodes just added to the tree, and possibly other nodes.

The topmost level of the algorithm is very simple, as shown in Fig. 5.6. Given the conceptual structure of the current instruction, it will create the corresponding individuals in the A-Box, setting a flag ACCOM if they don't exactly match known concepts; then it will call the procedure corresponding to the syntactic structure of the input instruction.

Needless to say, the algorithm is far from being fully developed: this is where my individual efforts and others' in the project come together. I consider myself responsible for the algorithms pertaining to purpose clauses and to negative imperatives. Clearly many other problems have to be addressed, for example temporal inferences, or plan recognition inferences of the kind that Kautz has proposed and that I described in sect. 2.

In the following, I present the algorithm for purpose clauses, and some ideas regarding the one for negative imperatives. My working assumption is that the various procedures receive in input a list of active nodes, so that they know in which subtree to insert the new structure derived from the current input instruction.

The algorithm for "Do α to do β"

Input: the individuals in A-box corresponding to Conceptual Structure representations of α and β, ACCOM, the current plan graph, and the list of active nodes.

1. Retrieve from the action library the simple plan(s) associated to β – definition(s) of which β is the header, generation relations in which β is the generatee, enablement relations in which β is the enablee.

2. If ACCOM is not set

   (a) If there is a direct *generation* or *enablement* relation between α and β, augment plan graph with the structure derived from it, after calling the procedure for computing possible assumptions (`compute-assumptions`).

   (b) If there is no such direct relation, recursively look for an indirect relation between the two actions, namely, look for possible connections between α and the substeps in the definitions for β – as I showed in sect. 5.2.3 for *go into the other room* and *get the urn of coffee.* This step may require calling the procedure for computing assumptions (`compute-assumptions`). Augment plan graph.

3. If ACCOM is set,

   (a) if there is GEN(ω, β, *agent*, t) or ENABLES(ω, β, *agent*, t), check whether

      i. ω is an ancestor of α: take α as the intended action;

      ii. ω is a descendant of α: take ω as the intended action.

      iii. ω and α are unifiable: take the unified description $\omega \sqcup \alpha$ as the intended action.

      iv. if ω and α are not unifiable because of disjoint role fillers, signal *failure*.

   (b) If there is no direct generation or enablement relation between α and β, proceed as in step 2b. Given that α is not known to the system, use heuristics similar to the ones employed in 3a in order to find the relations between α and the substeps of β.

A point I haven't addressed in the algorithm is what to do when step 1 yields more than one simple plan. In this case, the intended plan is the one which includes an action that matches α. For example, there could be more than one plan indexed by *get the urn of coffee*; *go into the other room* will help select those that include a physical movement on the part of the agent. If after this selection step there still is more than one available plan, different hypotheses can be pursued in parallel, or some measure of best "match" can be devised.

5.4 The algorithm for negative imperatives

So far, I have devoted far more attention to purpose clauses than to negative imperatives; therefore, I will simply make some comments on the inferences needed for negative imperatives, and on how the representation of the plan graph is affected by negative imperatives.

In the plan graph as defined so far, there is no provision for representing that an action should NOT be performed. Remember also the difference between *DONT* and *neg-TC* imperatives: in the former case the action α to be avoided is "independent", in the latter α is a side-effect or an undesirable way of performing another action β. The very first thing that comes to mind is to introduce two new arcs, *AVOID*, and *AVOID-IN*. In Figs. 5.7 and 5.8, these arcs are used to capture the difference between Exs. 45.a and .b, repeated here for convenience:

Ex. 52 a. *Use a vinyl-to-vinyl paste for any type of border you plan to hang over wallpaper. To paste the border for hanging, cover the entire back with paste and book the strip;* **don't crease the folds.**
b. *To book the strip, fold the bottom third or more of the strip over the middle of the panel, pasted sides together,* **taking care not to crease the wallpaper sharply at the fold.**

As far as *AVOID* is concerned, though, its meaning seems to amount to more than simply *avoid the action at the end of the arc*. In Ex. 52.a, what has to be avoided is performing *crease* on the result of *book*. If *crease* were executed, it would follow *book*: the AVOID arc has a temporal flavor to it. In the case of Ex. 23, relative to *cleaning the parquet*, the actions to be avoided, e.g. *scrub*, are alternative ways to generate the goal *clean the parquet*: the AVOID arc in this case should capture that *scrub* would generate *clean parquet*, if it were executed. As a first working hypothesis, I will then assume that the arcs *gen*, *temporal relations*, *substep*, *enables* have a corresponding AVOID arc – given that these is a very tentative solution, I am representing the AVOID arcs as dashed lines.

In ch. 4, I also mentioned, with respect to Ex. 23, that an algorithm similar to Kautz's could be used in this case, after adapting it to a representation based on GEN – in which we have for example

GEN(vacuum(agent, floor), clean(agent, floor), agent, t),
GEN(scrub(agent, floor), clean(agent, floor), agent, t)

A possible plan graph for a simplified version of Ex. 23 is shown in fig. 5.9.

5.5 Application: Animation from Instructions

The application system in which my work is grounded is the *Animation from Natural Language (AnimNL)* project at the University of Pennsylvania.

Over the years, Penn's Graphics Laboratory has developed an extensive model-based animation system. The system embodies anthropometric, kinematic and dynamic models, so that agents of different builds and strengths can be animated to perform tasks such as *grasp, look at, stand up, sit down* etc.

Given such model-based animation, it makes sense to envision a system where agents are given goals to achieve, or instructions to perform. Such a system could be used, among other things, to instruct human agents on how to perform a task; as an aid to designers, e.g. to check that the product is designed correctly for maintenance and repair; as an aid to instruction manuals writers, e.g. to ensure that their instructions are understandable.

Given the wide variety of possible users and applications for such a system, the most suitable and flexible language for interacting with the animated agent is Natural Language, as it is the only communication source accessible to users other than programming-wise animators [Badler *et al.*, 1990; Levison, 1991; Webber *et al.*, 1991].

Fig. 5.10 represents the architecture of the system. Referring to the figure, the part of diagram above the *action gate* represents the understanding and reasoning process an agent engages in prior to any commitment to action, and has as a result, a generation of such commitment.

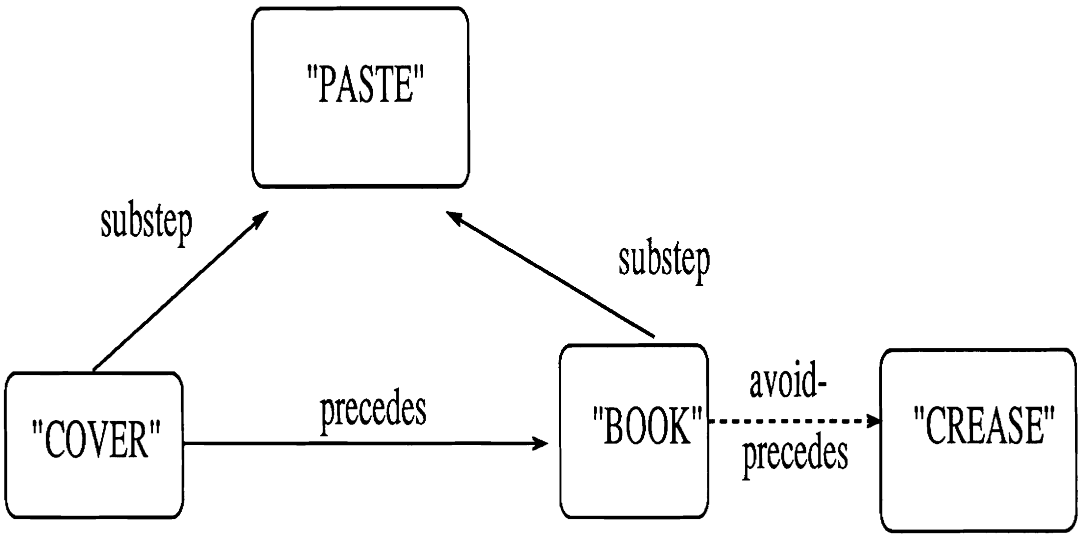

Figure 5.7: *Creasing* as an independent action

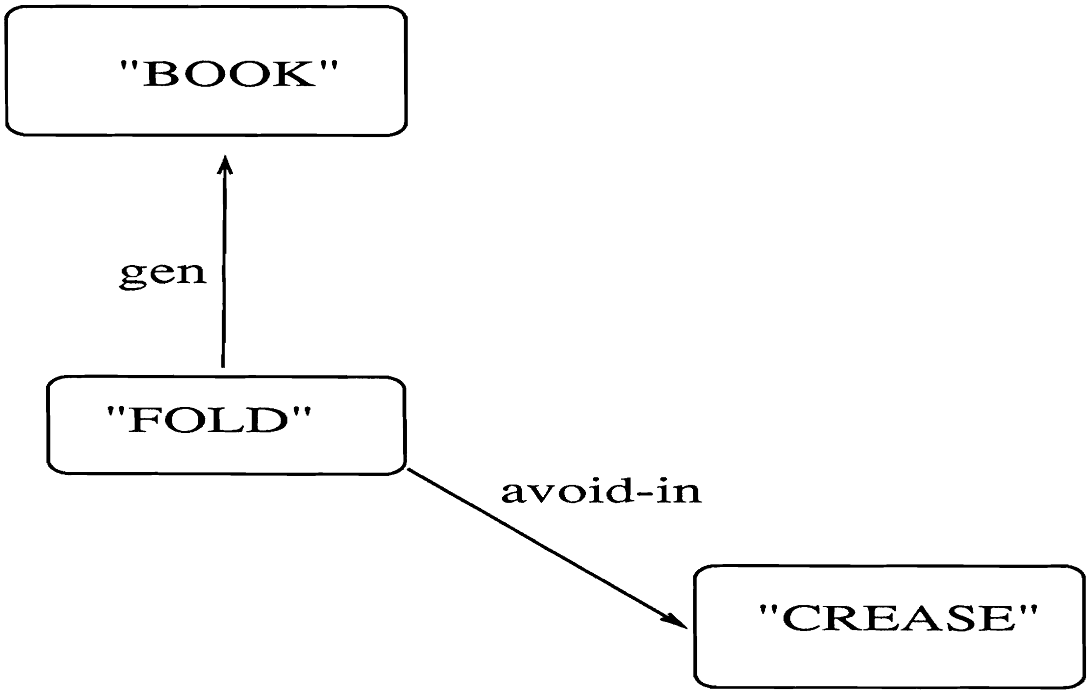

Figure 5.8: *Creasing* as a side-effect

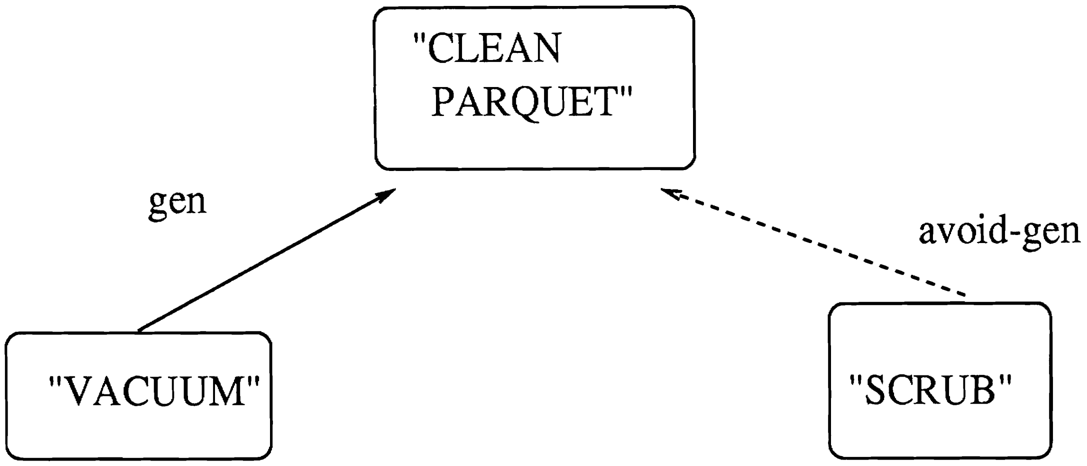

Figure 5.9: The *clean parquet* example

Instructions are given to AnimNL in *steps* consisting of one or more utterances. A step specifies a *continuous behavior* the agent must attend to - Ex. 51 is one such step. Steps are processed by a parser that uses a Combinatory Categorial Grammar [White, 1991], and produces a logical form in terms of Conceptual Structures. Such logical structures are incrementally developed into the plan graph, by means of processes of *plan inference, planning* and *plan grounding*. The inferences I discussed in this proposal mainly belong to the *plan inference* box.

Clearly, the nodes in the plan graph as shown in fig. 5.5 are not sufficiently specified to drive a simulator. First of all the plan needs to be ground into the particulars of the perceptually knowable environment: further planning will enable nodes of the plan graph to become more and more detailed.

When an intention becomes sufficiently specified for the agent to be ready to commit to it, the intention is gated, triggering another, low-level planning process - the part below the "action gate" in fig. 5.10. This low-level planning process takes care of postural planning, namely, of planning how the agent can bring himself in a position in which he can perform the requested action. Work being done by Jung [1991] is aimed at planning the postures needed in the course of satisfying an action. Finally, *animation directives* are sent to YAPS, the simulator – in fig. 5.10, Jack is the graphics system supporting animation.

5.6 Proposed work

Finally, I would like to summarize the work I am proposing to do on the model of instruction understanding.

- Theoretic part.
  - The speaker / hearer's model: extend and verify the axioms I wrote, and define various predicates such as CONSTRAINED, CONTRIBUTES, AVOID.
  - Give proper and accurate definitions for generation and enablement.

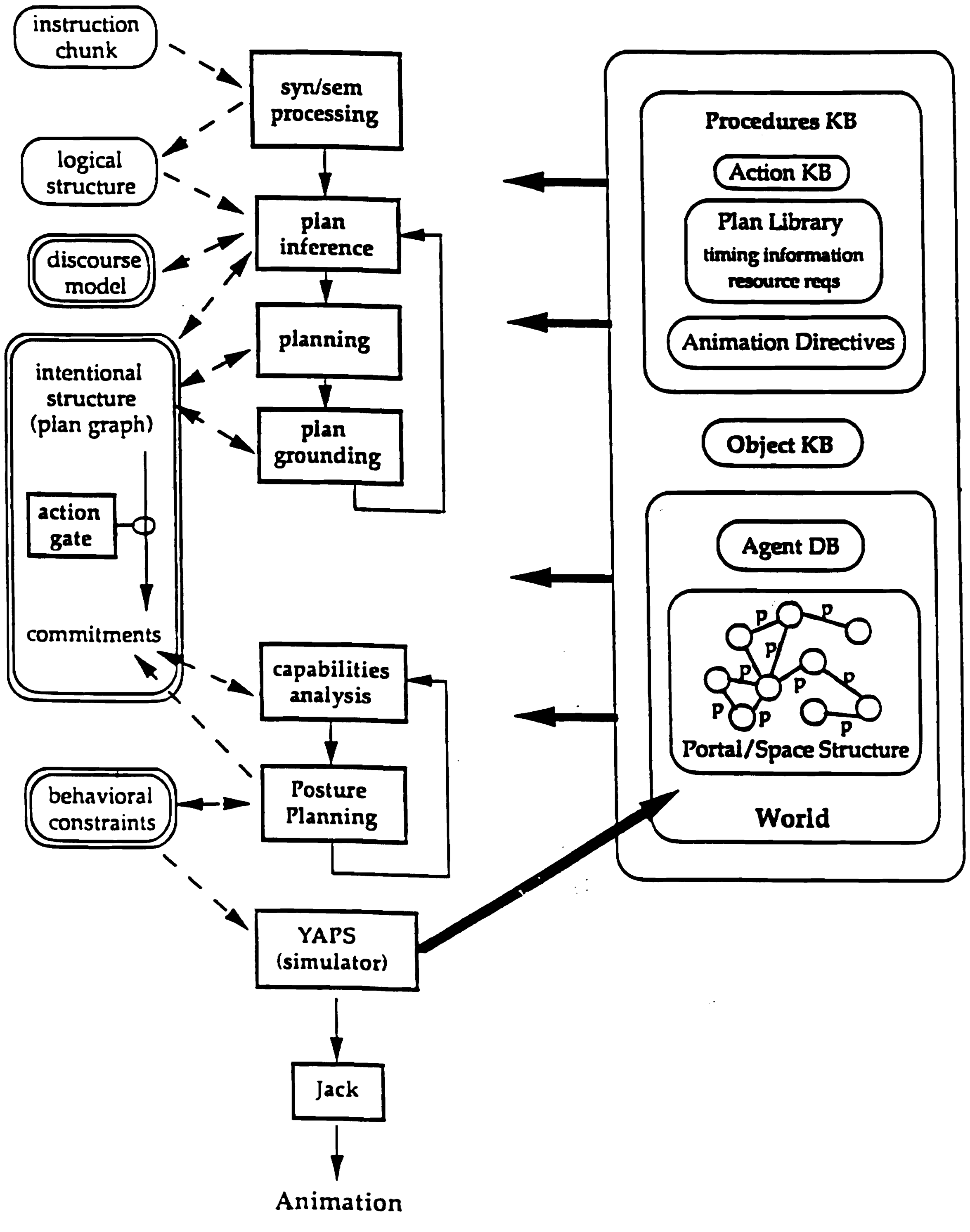

Figure 5.10: The architecture of the AnimNL system

  - Develop adequate inferences for negation.
  - Study how applicable the accommodation inferences are. Do the two types I developed with regard to purpose clauses – computing a more specific description of an action, and computing assumptions on the basis of which a given relation between two actions goes through — need to be integrated? may it happen that a whole structure composed of a goal and other actions needs to be integrated in the plan graph? do these inferences have broader application, for example to cases in which the goals are not explicitly given? How do the inferences necessary for negation relate to the ones developed for purpose clauses?

- Implementation.
  - Integrate hybrid systems and Conceptual Structures – I have been using the CLASSIC system [Brachman *et al.*, 1990] so far. The main question that needs to be addressed is whether the classification algorithm implemented in CLASSIC can deal with "multi-level" concept definitions.
  - Develop and implement a more precise algorithm to deal with purpose clauses and in particular with negative imperatives. An issue that needs to be dealt with is the updating of the list of active nodes in the plan graph. Another is the representation of actions that should not be executed in the plan graph: rather than having a single AVOID arc, the solution may be having different arcs, corresponding to the relation – e.g. generation, enablement etc – that the action α to be avoided would bear to another action in its context, if α were executed.

Chapter 6

Conclusions

The purpose of this proposal is to put forward the following claim:

most instructions don't exactly mirror the agent's knowledge, but are understood by accommodating them in the context of the general plan the agent is considering; the agent's accommodation process is guided by the goal(s) that s/he is trying to achieve. The concept of goal itself is pervasive in NL instructions, and a NL system which interprets instructions must be able to recognize and/or hypothesize goals, keep track of them, and use them in computing the description of the action to be performed.

I hope I managed to convince the reader that such claim is justified. The evidence I provided rests on the analysis of naturally occurring instructions, and in particular of positive imperatives containing purpose clauses, and of negative imperatives.

The analysis of purpose clauses has allowed me to show that goals do direct inference processes. I defined two inference processes pertaining to accommodation: computing a more specific action description and computing the expectations H builds upon hearing or reading a certain instruction.

The analysis of purpose clauses also has important consequences for the characteristics that an action representation formalism should have: computing more specific action descriptions requires that the formalism be flexible enough to recognize different levels of specificity; the relations between actions that purpose clauses embody are generation and enablement, that must therefore be included in the representation formalism.

The analysis of negative imperatives has shown that there are two classes of negative imperatives, *DONT* and *neg-TC* imperatives, that perform different pragmatic functions. They differ in the expectations that S has on the intentions H will adopt, and in the relation existing between the negated action and other actions in the context.

On the basis of the analysis of these data, I define a computational model of instructions consisting of three components, an axiomatic formalization of the communicative act taking place between S and H, an action representation formalism, and inference processes that build the structure of H's intentions.

More in detail,

1. The speaker / hearer model of imperatives is based on Cohen and Levesque's model,

which accounts for the intentions H adopts in response to S's utterance. I extended it by providing axioms that model purpose clauses, *DONT* and *neg-TC* imperatives.

2. The action representation formalism integrates hybrid knowledge representation systems and Conceptual Structures. I showed that the former are useful to compute one kind of accommodation, and the latter facilitates computing the other kind of accommodation. I showed that a T-Box can be easily expressed in Conceptual Structure terms.

3. I discussed the data structure adopted for the *plan-graph*, and sketched an algorithm that builds it.

A lot of work remains to be done, including:

- For the theoretic part,
  - Extend and verify the axioms I wrote to extend the speaker / hearer's model, and define various predicates such as CONSTRAINED, CONTRIBUTES, AVOID.
  - Give proper and accurate definitions for generation and enablement.
  - Develop adequate inferences for negation.
  - Study how applicable the accommodation inferences are. Do the two types I developed with regard to purpose clauses – computing a more specific description of an action, and computing assumptions on the basis of which a given relation between two actions goes through — need to be integrated? may it happen that a whole structure composed of a goal and other actions needs to be integrated in the plan graph? do these inferences have broader application, for example to cases in which the goals are not explicitly given? How do the inferences necessary for negation relate to the ones developed for purpose clauses?
- For the implementation,
  - Integrate hybrid systems and Conceptual Structures – I have been using the CLASSIC system [Brachman *et al.*, 1990] so far. The main question that needs to be addressed is whether the classification algorithm implemented in CLASSIC can deal with "multi-level" concept definitions.
  - Develop and implement a more precise algorithm to deal with purpose clauses and in particular with negative imperatives. An issue that needs to be dealt with is the updating of the list of active nodes in the plan graph. Another is the representation of actions that should not be executed in the plan graph: rather than having a single AVOID arc, the solution may be having different arcs, corresponding to the relation – e.g. generation, enablement etc – that the action α to be avoided would bear to another action in its context, if α were executed.

Acknowledgements

First and foremost, I want to thank my advisor, Bonnie Webber, for her insights, guidance and support. My work has greatly benefited from the AnimNL project meetings, and from discussions with its members, Norm Badler, Breck Baldwin, Chris Geib, Moon Jung, Libby Levison, Mike Moore, Mike White. Moreover, thanks to Mike Moore, Michael Niv, Christine Nakatani, Massimo Poesio, Lyn Walker, Mike White for discussions and feedback on this document. For financial support I acknowledge DARPA grant no. N0014-90-J-1863 and ARO grant no. DAAL03-89-C0031PR1.

Bibliography

[Allen, 1984] James Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[Allen, 1990] James Allen. Two Views of Intention: Comments on Bratman and on Cohen and Levesque. In J. Morgan, P. Cohen, and M. Pollack, editors, *Intentions in Communication*, MIT Press, 1990.

[Alterman *et al.*, 1991] Richard Alterman, Roland Zito-Wolf, and Tamitha Carpenter. *Interaction, Comprehension, and Instruction Usage.* Technical Report CS-91-161, Dept. of Computer Science, Center for Complex Systems, Brandeis University, 1991.

[Bach, 1990] Emmon Bach. *Informal Lectures on Formal Semantics.* SUNY Press, 1990.

[Badler *et al.*, 1990] Norman Badler, Bonnie Webber, Jeff Esakov, and Jugal Kalita. Animation from instructions. In Badler, Barsky, and Zeltzer, editors, *Making them Move*, MIT Press, 1990.

[Balkanski, 1990] Cecile Balkanski. *Modelling act-type relations in collaborative activity.* Technical Report TR-23-90, Center for Research in Computing Technology, Harvard University, 1990.

[Brachman and Schmolze, 1985] Ronald Brachman and James Schmolze. An overview of the KL-ONE Knowledge Representation system. *Cognitive Science*, 9:171–216, 1985.

[Brachman *et al.*, 1983a] R. Brachman, R.Fikes, and H. Levesque. *KRYPTON: A Functional Approach to Knowledge Representation.* Technical Report FLAIR 16, Fairchild Laboratories for Artificial Intelligence, Palo Alto, California, 1983. Also in *Readings in Knowledge Representation*, Brachman, Levesque eds., Morgan-Kaufmann, 1985.

[Brachman *et al.*, 1983b] R. Brachman, R.Fikes, and H. Levesque. KRYPTON: Integrating Terminology and Assertion. In *Proceedings AAAI 83*, 1983.

[Brachman *et al.*, 1990] Ronald Brachman, Deborah McGuinness, Peter Patel Schneider, Lori Alperin Resnick, and Alexander Borgida. Living with CLASSIC: when and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks*, Morgan Kaufmann Publishers, Inc., 1990.

[Brachman *et al.*, 1979] R. Brachman, R. Bobrow, P. Cohen, J. Klovstad, B. Webber, and W. Woods. *Research in natural language understanding, annual report.* Technical Report 4274, Bolt Beranek and Newman, 1979.

[Chapman, 1991] David Chapman. *Vision, Instruction and Action.* Cambridge: MIT Press, 1991.

[Cohen and Levesque, 1990a] Philip Cohen and Hector Levesque. Persistence, Intention and Commitment. In J. Morgan, P. Cohen, and M. Pollack, editors, *Intentions in Communication*, MIT Press, 1990.

[Cohen and Levesque, 1990b] Philip Cohen and Hector Levesque. Rational Interaction as the Basis for Communication. In J. Morgan, P. Cohen, and M. Pollack, editors, *Intentions in Communication*, MIT Press, 1990.

[Cohen and Levesque, 1991] Philip Cohen and Hector Levesque. Teamwork. 1991. Manuscript.

[Di Eugenio and White, 1991] Barbara Di Eugenio and Michael White. On the Interpretation of Natural Language Instructions. 1991. Submitted to COLING 92.

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

[Goldman, 1970] Alvin Goldman. *A Theory of Human Action.* Princeton University Press, 1970.

[Halpern and Moses, 1985] J.Y. Halpern and Y.O. Moses. A guide to the modal logics of knowledge and belief. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985.

[Hegarty, 1990] Michael Hegarty. Secondary Predication and Null Operators in English. 1990. Manuscript.

[Heim, 1982] Irene Heim. *The Semantics of Definite and Indefinite Noun Phrases.* PhD thesis, University of Massachussets, 1982.

[Jackendoff, 1983] Ray Jackendoff. *Semantics and Cognition. Current Studies in Linguistics Series*, The MIT Press, 1983.

[Jackendoff, 1990] Ray Jackendoff. *Semantic Structures. Current Studies in Linguistics Series*, The MIT Press, 1990.

[Jones, 1985] Charles Jones. Agent, patient, and control into purpose clauses. In *Chicago Linguistic Society, 21*, 1985.

[Jung, 1991] Moon Jung. Posture Planning for Human Task Animation in Workspaces. 1991. Technical Report, CIS Dept, Univ. of Pennsylvania. To appear.

[Kautz, 1990] Henry Kautz. A circumscriptive theory of plan recognition. In J. Morgan, P. Cohen, and M. Pollack, editors, *Intentions in Communication*, MIT Press, 1990.

[Lambert and Carberry, 1991] Lynn Lambert and Sandra Carberry. A tripartite plan-based model of dialogue. In *Proceedings of the 29th Annual Meeting of the ACL*, pages 47–54, 1991.

[Levison, 1991] Libby Levison. *Action Composition for the Animation of Natural Language Instructions.* Technical Report MS-CIS-91-28, University of Pennsylvania, 1991.

[Lewis, 1979] David Lewis. Scorekeeping in a language game. *Journal of Philosophical Language*, 8:339–359, 1979.

[Litman and Allen, 1990] Diane Litman and James Allen. Discourse processing and commonsense plans. In J. Morgan, P. Cohen, and M. Pollack, editors, *Intentions in Communication*, MIT Press, 1990.

[Lochbaum, 1991a] Karen Lochbaum. *Plan recognition in collaborative discourse.* Technical Report TR-14-91, Center for Research in Computing Technology, Harvard University, 1991.

[Lochbaum, 1991b] Karen Lochbaum. An algorithm for plan recognition in collaborative discourse. In *Proceedings of the 29th Annual Meeting of the ACL*, pages 33–38, 1991.

[Moore, 1980] R. Moore. *Reasoning about Knowledge and Action.* Technical Report 191, Artificial Intelligence Center, SRI International, 1980.

[Pollack, 1986] Martha Pollack. *Inferring domain plans in question-answering.* PhD thesis, University of Pennsylvania, 1986.

[Suchman, 1987] Lucy Suchman. *Plans and situated actions.* Cambridge University Press, 1987.

[Vere and Bickmore, 1990] Steven Vere and Timothy Bickmore. A basic agent. *Computational Intelligence*, 6:41–60, 1990.

[Vilain, 1985] Mark Vilain. The Restricted Language Architecture of a Hybrid Representation System. In *IJCAI-85*, 1985.

[Webber and Di Eugenio, 1990] Bonnie Webber and Barbara Di Eugenio. Free Adjuncts in Natural Language Instructions. In *Proceedings Thirteenth International Conference on Computational Linguistics, COLING 90*, pages 395–400, 1990.

[Webber *et al.*, 1991] Bonnie Webber, Norman Badler, Barbara Di Eugenio, Libby Levison, and Michael White. Instructing Animated Agents. In *Proc. US-Japan Workshop on Integrated Systems in Multi-Media Environments. Las Cruces, NM*, 1991.

[White, 1991] Michael White. Conceptual Structures and CCG: Linking Rules and Incorporated Argument Adjuncts. 1991. Submitted to COLING '92.

[Winograd, 1972] Terry Winograd. *Understanding Natural Language.* Academic Press, 1972.

[Zwarts and Verkuyl, 1991] Joost Zwarts and Henk Verkuyl. An algebra of conceptual structure; an investigation into Jackendoff's conceptual semantics. 1991. Accepted for publication in *Linguistics and Philosophy.*