Generating Referring Expressions in a Domain of Objects and Processes

Robert Dale

PhD

University of Edinburgh

1988

Declaration

I declare that this thesis has been composed by myself and that the research reported therein has been conducted by myself unless otherwise indicated.

Robert Dale

Edinburgh, 27th November 1988

Acknowledgements

A great many people have contributed, in one way or another, to the completion of this thesis.

Special thanks to: Judy Delin, who suffered most, and managed to keep me sane; Ewan Klein for being a friend as well as a supervisor; and Jon Oberlander for always being willing to listen to my half-baked ideas.

Ewan Barker, Jo Calder, Richard Caley, Peggy Coonagh, George Dunbar, Frank van Harmelen, Betty Hughes, Colin Matheson, Marc Moens (24), David Moffat, and Mike Reape all helped with little things at one time or another; Nick Haddock, Kate Stainton-Ellis, Mary Tait and William the cat each influenced my life and thesis (at times indistinguishable) in subtle ways they probably won't ever be aware of; and Bonnie Webber provided encouragement just when I needed it most. Ewan Klein and Graeme Ritchie provided supervision during the lengthy period in which the thesis took shape, reading almost as many drafts as they've had hot dinners. Thanks are also due to Peter Hornsby of Syntek Limited, who, during the final two years of thesis preparation, funded research which was not related to the thesis work but provided me with an environment and the financial security that makes all the difference; and to Gerard Kempen and Chris Mellish, my thesis examiners.

Finally, thanks to everyone on board the Starship Epistemics for contributing to a great working environment, and especially to Ewan, Marc and Jon for fetching the croissants.

Abstract

This thesis presents a collection of algorithms and data structures for the generation of pronouns, anaphoric definite noun phrases, and *one*-anaphoric phrases. After a close analysis of the particular kinds of referring expressions that appear in a particular domain—that of cookery recipes—the thesis presents an appropriate ontology and a corresponding representation language. This ontology is then integrated into a wider framework for language generation as a whole, whereupon we show how the representation language can be successfully used to produce appropriate referring expressions for a range of complex object types.

Amongst the more important ideas explored in the thesis are the following:

- We introduce the notion of a generalized physical object as a way of representing singular entities, mass entities, and entities which are sets.
- We adopt the view that planning operators are essentially underspecified events, and use this, in conjunction with a simple model of the hearer, to allow us to determine the appropriate level of detail at which a given plan should be described.
- We make use of a discourse model that distinguishes local and global focus, and is closely tied to a notion of discourse structure; and we introduce a notion of DISCRIMINATORY POWER as a means to choosing the content of a referring expression.
- We present a model of the generation of referring expressions that makes use of two levels of intermediate representation, and integrate this model with the use of a linguistically-founded grammar for noun phrases.

The thesis ends by making some suggestions for further extensions to the work reported here.

Notational Conventions

In this thesis, the following notational conventions are used.

Example of Convention	Description of Convention
a linguistic example	Linguistic examples in the body of the text are in italics.
A displayed example	However, in displays, linguistic examples appear in roman text.
This is an example	In displayed examples, the particular item of interest (usually an anaphor or a noun phrase) appears underlined.
the block which is on itself	Wherever relevant in displayed examples, the antecedents of anaphors appear in italics.
...	A sequence of three dots is used to indicate ellipsis, i.e., where some material not currently of interest has been excised from an example.
Mike had dahl and Jo ϕ pakora	The symbol ϕ is used to indicate that an element is assumed to have been elided from a syntactic structure.
*The skunks is friendly	A sentence or phrase which is ungrammatical is denoted by a preceding '*'.
?I haven't an oat	A sentence or phrase whose well-formedness is questionable is denoted by a preceding '?'.
#The dog permuted his tail-lights	A sentence or phrase which is grammatical but unacceptable in the given context is denoted by a preceding '#'.
$\forall x F(x) = G(x)$	Italics are also used for mathematical and logical symbols.
WIDGET	Small capital letters are used to indicate that a technical term is being used for the first time.
SCRUMPY	Small capital letters are also used to indicate the names of programs.
`foo(bar)`	A typewriter-like typeface is used to display program segments.

A Note on Citations

In citations, the year of the citation refers to the year of publication of the work; however, in many cases, the work itself is more accessible in a more recent publication. A common example of this is Frege's [1892] 'On Sense and Reference', which is much more widely available in more recently published collections than in the original. In such cases, both the original source and the reprint are indicated in the bibliography, although the

citation will refer to the earlier work: thus, the text will contain the citation form 'Frege [1892]' rather than 'Frege [1980]'.

Following Zwicky's *Second Law of Reference* [Zwicky 1986:122-123], I have tried to make references to publications as specific as possible: by page numbers, section numbers, or chapter numbers, whichever is most appropriate or helpful. Where the date of a citation refers to a more inaccessible edition, such page numbers generally refer to the more accessible publication: thus, 'Frege [1892:69]' really means page 69 of the 1980 collection within which Frege [1892] was reprinted. Where the bibliography lists both an original work and a more recent reprint, the particular source the page numbers refer to is indicated.

Chapter 1

Introduction

Any natural language generation system has to have the ability to talk about things; and in order to talk about things, it has to be able to construct referring expressions. This thesis describes EPICURE, a natural language generation system whose principal concern is the generation of REFERRING EXPRESSIONS which pick out COMPLEX ENTITIES in connected DISCOURSE. Roughly speaking,[1] by a referring expression we mean a linguistic form which is used to pick out an entity in some real or imaginary world; by complex entities we mean not only singular individuals such as the referents of simple noun phrases like *a cat* and *the mat*, but also what philosophers of language would call non-singular individuals, that is, entities which are described by mass and plural noun phrases such as *some cheese* and *a thousand sneezes*; and we take a discourse to be a coherent stretch of natural language larger than a single sentence or utterance. This last concept allows us to place particular emphasis on the generation of ANAPHORIC referring expressions: informally, an anaphoric referring expression is an abbreviated linguistic form, where the entity picked out by that expression can only be determined by making use of contextual information, rather than from the content of the linguistic form itself. So, anaphoric referring expressions are typically used to refer to entities which are in some way known to the hearer, usually because they have been referred to already, or because they are somehow present in the environment, either because their existence is already assumed, or because they are actually physically present.

In order to support the generation of referring expressions, this thesis also describes a language generation system which provides the appropriate infrastructure. EPICURE is

[1] The precise meanings of these terms intended in the present work will become clearer as the thesis unfolds.

designed to generate connected discourse from specifications of plans of action. In this thesis, the particular plans considered are cookery recipes, although the general ideas worked out here should be straightforwardly applicable to plans constructed in other domains. By accessing a database of information about the plans of action that the hearer is assumed to know how to perform, EPICURE decides what level of description is required for the hearer to understand the actions in the plan. A description of each action then forms the basis of an utterance to be generated. For each entity to be mentioned in each such utterance, EPICURE generates an appropriate referring expression, taking into account both the structure and content of the discourse so far, and also what the hearer can be assumed to know on the basis of information external to the discourse. Thus, by modifying the system's knowledge of the hearer's knowledge, different discourses may be generated.

The rest of this chapter situates the work in the field of language generation as a whole. After first stating a little more precisely the phenomena to be considered in the thesis, we go on to summarise the aims and contributions of the thesis, and to state the starting points and methodology adopted in the execution of the work described. The chapter ends by providing a brief overview of the operation of EPICURE.

1.1 The Phenomena Considered

Much has been written on the nature of anaphora. Some of this work will be reviewed in detail in chapter 2, but it will be useful at this point to provide a little more insight into the particular phenomena considered in this thesis.

Our principal concern is the generation of three particular kinds of anaphora: PRONOMINAL ANAPHORA, DEFINITE NOUN PHRASE ANAPHORA, and ONE-ANAPHORA, as exemplified by (1-1), (1-2) and (1-3) respectively.

(1-1) a *A cat* walked into the room.
b It was wearing a red collar.

(1-2) a *A cat* and a dog walked into the room.
b The cat was wearing a red collar.

(1-3) a A cat and a dog walked into the room.

b The cat was wearing *a red collar*, and the dog was wearing a blue one.

Natural language makes available a number of other anaphoric forms, but the generation of these three types constitutes the primary focus of the thesis.[2] In particular, the thesis examines the following practical decisions that need to be made by any system which is to be considered sophisticated in this area:

- When can a pronoun be used to refer to an entity?
- When can a definite determiner be used in referring to an entity?
- How is the semantic content of a referring expression decided upon?
- When can *one*-anaphora be used to refer to an entity?

These questions, in turn, give rise to more fundamental philosophical questions:

- What does it mean to 'refer' to something?
- What kinds of things are 'entities'?

An attempt will be made to answer both the practical and philosophical questions, within the context of generating discourses from plans.

Of course, there are many aspects of the generation of referring expressions *not* touched upon in the present work. In particular, the thesis does not attempt to cover spoken language, or deixis (i.e., language used to identify extralinguistic referents, perhaps in conjunction with implicit or explicit pointing): it is only concerned with the written form, as befits machines which are not yet too comfortable either speaking or pointing. There are, of course, similarities between spoken and written language, and between deictic and non-deictic uses of language, and many of the ideas discussed in this thesis may have some bearing on these other forms of language use; however, the differences are sufficiently complex to warrant the more specific focus adopted here. Finally, the thesis is concerned only with the resources available in a single natural language, namely English. This is not because of any uniquely interesting characteristics of that language, but solely due to my ignorance of any other language in sufficient detail.

[2] See Hirst [1981b] for an excellent survey of the range of different types of anaphoric expressions.

1.2 The Aims of the Work

The work described in the thesis has aims on two levels. The more general aims of the work are of a methodological nature, and are best considered against the background provided by two observations on the current state of the field of natural language generation.

First, there is a need for consolidation in the field. For a long time, research in natural language generation has been the 'poor relative' of work in natural language understanding: many more researchers have been attracted to the problems posed by the latter than to those posed by the former. The situation is no longer as extreme as it was, and recent years have seen the more widespread recognition of work on issues in language generation: the most widely cited works in this area are those of Davey [1978], McDonald [1980a], McKeown [1982, 1985], and Appelt [1982, 1986]. Still, it is very noticeable that, whereas there is now a well-grounded body of theory available for natural language understanding (for example, in the area of parsing techniques), there is no such comparable consolidation of results within the work on language generation. In fact, it is often unclear as to whether different systems are addressing the same questions.

Second, sufficient work has now been done in the field for it to be the case that the days when a doctoral thesis could present yet another generation system are past; research in language generation must now become more oriented towards in-depth analysis of particular aspects of the generation task, building on what has gone before. Here there is a problem: as anyone surveying the field quickly discovers, it is very difficult to build upon what has gone before, largely because the foundations have not yet been put in place. A major problem, for example, is the lack of agreement amongst researchers as to what the input to the generation process should be, and so each system makes its own assumptions and develops its own *ad hoc* techniques for solving the particular problems that arise as a result of those assumptions being made.

Given these observations, this thesis attempts to consolidate and build upon the work already done on one particular problem in language generation: that of REFERRING. The work extends considerably the ideas presented in the major works mentioned above, and also those of a number of other researchers. Apart from this desire to consolidate, the work is also driven by a desire to be as generally useful as possible. Given the

current lack of agreement with regard to choices of formalism and representation, the work presented tries to be as explicit and objective about its assumptions as possible. Some first steps are made towards the development of a representation language that supports the distinctions relevant for the purposes of generating the wide variety of referring expressions to be found in natural language. Of course, it is too much to hope that the algorithms and data structures presented in this work can be picked up and used 'off-the-shelf' by others working on generation systems; but I hope to have presented the information in a way that makes it easy to see what changes would be required to accommodate different assumptions.

Of course, the thesis also has aims of a more specific nature. At the level of the particular issues it addresses, the major aim is to provide computational solutions to the generation of the phenomena briefly described in the previous section; that is, it aims to present a collection of algorithms and data structures for the generation of pronouns, anaphoric definite noun phrases, and *one*-anaphoric phrases. The mechanisms developed for this pay specific attention to the use of both discourse modelling and hearer modelling.

In summary, the aims of this thesis are

- to acknowledge the variety and complexity of the kinds of things that can be referred to, and to make some first steps towards the development of a representation language that can encompass this variety and complexity;
- to present a collection of algorithms and data structures for the generation of pronouns, anaphoric definite noun phrases, and *one*-anaphoric phrases using this representation language as a base; and, while doing so,
- to consolidate and advance upon what has been done in the area of referring expression generation by earlier researchers.

1.3 The Contributions of the Thesis

As already suggested, the principal contributions of this thesis lie in its attempt to consolidate and build upon existing work relating to one particular problem in language generation, in such a way as to enable others to make use of the research. At a more detailed level, the original contributions of the thesis lie in its presentation of specific

algorithms for the generation of pronominal anaphora, definite noun phrases, and *one*-anaphora. Most existing generation systems have something to say about the first and second of these, but rarely have anything to say about the third. In the case of pronominal anaphora, this thesis posits a fairly simple solution which crucially depends upon the notion of discourse model adopted. In the case of definite noun phrases and *one*-anaphoric phrases, the algorithms presented here are substantially more complex than those presented in any work extant at the time of writing, although they are developed from existing work in other areas of computational linguistics.

Apart from the linguistic coverage of the work presented here, there are two other themes that contribute to the originality of the thesis.

First, in order to provide generation algorithms that are as generally applicable as possible, the thesis adopts and develops a more complex and principled underlying ontology than is to be found elsewhere. This results in an approach to the representation of complex entities which permits a wider coverage in terms of noun phrase structures than would otherwise be possible.

Second, the system described makes explicit use of discourse structural information, in an explicit discourse model. The need for work on the notion of discourse modelling in language generation was identified as a crucial area in the 1982 ACL panel on text generation [Mann *et al* 1982], and that requirement is taken to heart here. In so doing, the thesis also makes use of some recent work on the structure of discourse by Grosz and Sidner [1985, 1986]; we make some specific observations in this regard that contribute to the simplicity of the approach taken to pronominalization.

1.4 Starting Points

The work reported here adopts particular stances on certain issues, and takes particular themes as starting points. The most important of these are as follows.

1.4.1 Methodology

This thesis is, in the first instance, an exercise in computational linguistics. Unlike some other work in natural language generation (for example, McDonald [1979, 1980a, 1980b] and Kempen and Hoenkamp [1982]), the present work does not make any claims for the psychological reality of the results obtained; similarly, it does not claim to unearth any deep linguistic truths. However, it is informed by research in psychology and linguistics: it is necessary to have some psycholinguistically-informed idea of the hearer's memory constraints, for example, in order to generate expressions which the hearer will understand; similarly, there are useful generalizations and observations to be gleaned from work in linguistics, without which the coverage of any system would be greatly impoverished. Thus, the overall aim is for an engineering solution to the problem of referring expression generation which is still sound from the point of view of linguistic theory, and cognisant of the relevant data from psycholinguistics.

1.4.2 The Domain of Application

The richness and breadth of natural language means that any attempt at a computational treatment has to narrow its focus in various respects. Apart from concentrating on particular linguistic phenomena, it is usual to also concentrate on a particular domain of application. In this thesis, the chosen domain can be most generally described as the description of plans of action: in particular, however, we focus on the description of cookery recipes. Cookery recipes usually look something like that shown in figure 1.1.[3]

The cookery recipe domain is an interesting domain for a number of reasons. It is fairly well-defined, and yet exhibits some remarkably complex phenomena; it comes as no surprise, then, to see that it has been chosen as a domain of investigation by a number of researchers. Lehrer [1969, 1972] has examined the vocabulary of cooking terms in depth; Hammond [1986] uses recipe construction as a domain for the investigation of planning techniques, and Tsang [1986] discusses temporal aspects of planning in cookery recipes; and most recently, Karlin [1988] has described work on a natural-language interface to a computer-generated animation system that operates in the domain of cooking.

[3]From *The Vegetarian Epicure*, Anna Thomas, Penguin 1972. It should be emphasised that this recipe is beyond the scope of anything EPICURE can currently generate.

HOT STUFFED MUSHROOMS

16 to 20 large mushrooms
2 dozen pitted ripe olives
8 to 10 pickled cocktail olives
$\frac{1}{2}$ pint strong vegetable broth
breadcrumbs, as needed
salt and pepper
butter

Wash the mushrooms, remove their stems, and scoop out a little of the insides. Chop the stems well and sauté lightly in a little butter. Mince the olives and onions and add them to the chopped mushrooms, along with a few tablespoons of the broth. Now add some breadcrumbs until the mixture is thick and stiff enough to handle. Season it with salt and pepper.

Stuff the mushroom caps with this mixture, arrange them in a baking dish, and pour the rest of the stock over them. Put a dab of butter here and there among them and bake in a 350°F oven (gas 4) for about 15 minutes. Serve piping hot.

Figure 1.1: Hot Stuffed Mushrooms

From the point of view of the research described in this thesis, the description of cookery recipes is an interesting domain for two principal reasons: first, it allows us to examine some of the issues that arise in generating discourses whose structure mirrors that of the plans that underly them; and second, it permits us to make use of a sophisticated ontology that encompasses more than just simple entities. The results of the research, however, should be applicable in many related domains, particularly those that have an instructional element and those which require a more complex underlying ontology than is normally found in work in computational linguistics.

1.4.3 What to Say vs How To Say It

There is a fairly standard distinction made in the generation literature between the decisions of *what to say* and *how to say it*.[4] The decisions involved in determining what to say are usually referred to as CONCEPTUAL or STRATEGIC, and the decisions involved in determining how to say something are usually referred to as LINGUISTIC or TACTICAL.

Most well-known work in generation adopts the distinction; thus, McDonald's MUMBLE [McDonald 1980a] is a tactical component which assumes the existence of a strategic component in the form of an 'expert speaker' that provides some specification of what is to be said. McKeown's TEXT [McKeown 1982, 1985], on the other hand, is a strategic component which produces utterance specifications that are passed to a separate tactical component for realization.[5]

From the point of view of implementing a working computer model of language generation, the distinction is a very useful one, allowing as it does the modularisation of the task of language generation into two distinct components. However, there are researchers working in the field of language generation who do not accept the validity of the distinction. Appelt [1982:6–8, 1986:6–8], for example, sees the distinction as being based on what Reddy [1979] has called the CONDUIT METAPHOR: i.e., the view that language is a pipe or conduit through which a speaker sends to a hearer some conceptual material wrapped in a package of natural language. Appelt's work, in contrast, views

[4] To my knowledge, the distinction was first discussed in detail by Thompson [1977], although the distinction itself is surely older. Thompson's motivation for the distinction is that 'dividing things up in this way makes it easy to describe the resulting parts and their interactions' [1977:653].

[5] Work by Rubinoff [1986] integrates McKeown's TEXT system and McDonald's MUMBLE.

decisions about 'what to say' and 'how to say it' as two aspects of the same process, and permits interactions between them.

Another recent detractor is Danlos [1984, 1987]. Although her system acknowledges distinct strategic and tactical components [Danlos 1987:122], the strategic component makes both conceptual and linguistic decisions: her claim is that conceptual and linguistic decisions cannot be made independently of one another [Danlos 1984:501]. She cites Appelt in her discussion of the distinction between 'what to say' and 'how to say it' [Danlos 1987:32–36], but on closer inspection her criticism of the distinction seems to be somewhat different from Appelt's. Discussing the question of the extent to which two sentences which describe the same event but embody different lexical and syntactic choices can be said to have the same meaning, she suggests that

> ...it is a question of deciding whether the way people or objects are referred to is part of the 'how to say it' question ...or part of the 'what to say' question ...
>
> [Danlos 1987:34]

Thus, although both Appelt and Danlos dispute the validity of the strategy/tactics distinction, they do so for subtly different reasons: for Appelt, there are two categories of decisions but they interact with each other; for Danlos, it is not obvious that it makes sense to distinguish two categories of decisions.

My own view is closer to that of Danlos. In the work described here, I take the view that 'what to say' and 'how to say it' are two sides of the same question: thus, the generator may be given the 'what to say' goal of requesting the hearer to make chestnut soufflé, and decide that 'how to say it' requires describing a particular plan of action; each such step in the plan can then be viewed as a specification of what to say, for which the generator has to decide how to say it. This recursion continues until an utterance is produced. To put it another way, a decision which, from the outside, might be categorised as a 'how to say it' decision, from the inside might be best described as a 'what to say' decision. Thus, choosing *how* a particular entity should be referred to involves choosing *what* semantic content the description should contain.

Note that, in a computational system, this approach does not preclude the use of distinct components for the different levels of the process, which simply call each other

recursively. So, we can both accept the arguments of those who say the distinction, at least as originally stated, is suspect; and at the same we can adopt the distinction as a useful conceptual viewpoint.

1.4.4 Planning and Language Generation

There are two ways in which the AI notion of planning[6] connects with work in language generation. First, there are language generation systems whose purpose is to describe plans of action as might be produced by a planning system: in such cases, the language generation program is basically a tactical component, assigned the task of determining how to say what the planner has specified has to be said. A good example of this approach can be found in recent work by Mellish [1988].

The second relationship between language generation and planning is more fundamental, however: this is the view that language itself is planned behaviour, and so can be modelled using the same tools and techniques developed for other kinds of planning. This view sees linguistic acts as being on a par with physical acts, and is founded on speech act theory (see, for example, Austin [1962] and Searle [1969]). Within computational linguistics, work in this area was first suggested by Bruce [1975a] and further developed by Allen, Perrault, Cohen and Levesque [Cohen 1978, Perrault and Allen 1978, Allen and Perrault 1978, Cohen and Perrault 1979, Perrault and Allen 1980, Cohen and Levesque 1980, Cohen 1981, Allen 1983].

In a manner similar to Appelt's work, the present work is intended to integrate both uses of planning, so that the behaviour that the system plans is the linguistic description of plans of action.[7]

1.4.5 Discourse Structure

Another crucial starting point adopted in the current work is a notion of DISCOURSE STRUCTURE derived from the work of Grosz and Sidner [1985, 1986]. Under this approach, a discourse can be viewed as consisting of distinct discourse segments, which

[6] Tate [1985] provides a good overview of the issues that arise in planning.

[7] Unlike Appelt's work, however, the present work does *not* include a full-blown planning mechanism; only limited reasoning about action is carried out by the system itself, as will be made clear in chapter 4.

bear both hierarchical and ordering relations to each other. The structure of the discourse that results from these relations plays a role in the construction of anaphoric expressions, and is used in the present work to provide a basis for the approach taken to discourse modelling.

1.4.6 Ontology and Levels of Representation

A major element of the work presented here is the recognition that, in the real world, entities are considerably more complex than those found in toy domains such as those consisting of nothing more than blocks and pyramids. The chosen domain presents interesting problems with respect to the representation of objects which can be countable or mass, which can be decomposed, and which can be collected together to create objects which did not exist previously. The distinctions we model at this ontological level are then mirrored appropriately in the two intermediate levels of representation used in the generation process: various aspects of this approach are similar in concept to the work of Webber [1979].

1.4.7 Formal Grammar

The 1982 ACL panel on language generation referred to above [Mann *et al* 1982] noted several areas where attention needed to be focussed in the development of language generation systems. One such area is the use of linguistically well-founded grammatical formalisms: until now, most work in language generation has made use of rather *ad hoc* approaches to grammatical description. The present work makes use of ideas central to unification-based approaches to grammar,[8] and includes a reasonably-sized grammar fragment for the description of noun phrases in English which has its roots in generalized phrase structure grammar (GPSG) [Gazdar *et al* 1985].

1.5 An Overview of the System

The system described in the rest of this thesis is presented as a diagram in figure 1.2,

[8] See Shieber [1986] for a good introduction.

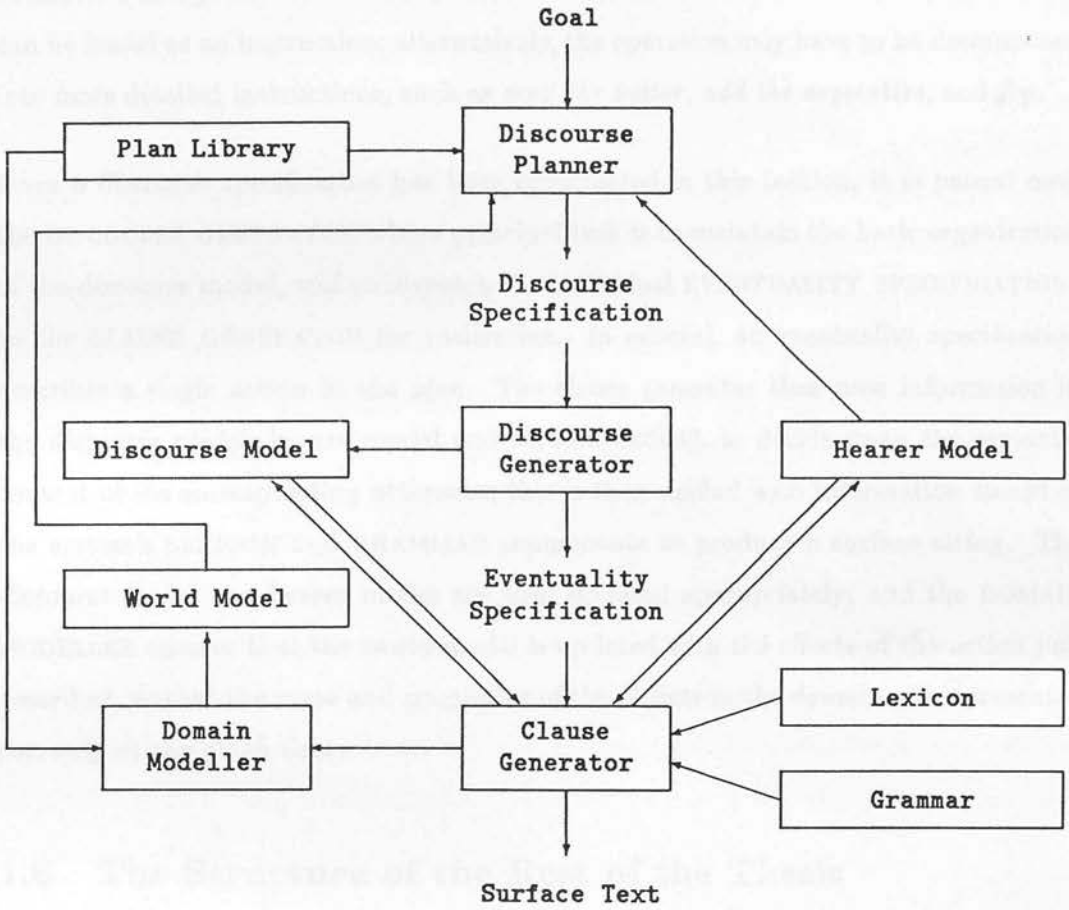


Figure 1.2: The Overall Structure of EPICURE.

and operates as follows.[9]

EPICURE is presented with a top level goal that is to be achieved. The DISCOURSE PLANNER component uses its PLAN LIBRARY in conjunction with its knowledge of what operations the hearer knows how to perform (as maintained in the HEARER MODEL) to produce a hierarchically-structured DISCOURSE SPECIFICATION which specifies the actions that must be requested of the hearer in order to achieve the goal state. As an example, if the system believes that the hearer understands what it means to *sauté*, this can be issued as an instruction; alternatively, the operation may have to be decomposed into more detailed instructions, such as *melt the butter*, *add the vegetables*, and *fry*.

Once a discourse specification has been constructed in this fashion, it is passed onto the DISCOURSE GENERATOR whose principal task is to maintain the basic organization of the discourse model, and to dispatch the individual EVENTUALITY SPECIFICATIONS to the CLAUSE GENERATOR for realization. In general, an eventuality specification describes a single action in the plan. The clause generator then uses information in the discourse model, hearer model and WORLD MODEL to decide upon the semantic content of the corresponding utterance; this is then unified with information stored in the system's LEXICON and GRAMMAR components to produce a surface string. The discourse model and hearer model are then updated appropriately; and the DOMAIN MODELLER ensures that the world model is updated with the effects of the action just described, so that the state and properties of the objects in the domain are represented correctly at any given time.

1.6 The Structure of the Rest of the Thesis

The rest of this thesis is structured as follows.

Chapter 2 looks at the existing literature on reference and anaphora. The chapter begins by discussing briefly the notion of reference adopted in this thesis, before going on to describe the various forms of reference available in the English language, and the various anaphoric devices available. Typically, the entity referred to by an anaphoric

[9] In this diagram, data structures are represented by long flat boxes, whereas process modules are represented by the more squat boxes. An arrow pointing from box A to box B is intended to show that data flows from A to B.

expression will have been referred to elsewhere in the text under consideration. Adopting standard terminology, we call the previous linguistic form used to refer to the entity the ANTECEDENT; an anaphoric expression and its antecedent are said to CO-REFER, i.e., pick out the same entity in the world. We look at several important elements of anaphora, considering, in turn, the nature of the referent, the relative location of the antecedent in the text, and the relationship between the anaphor and its antecedent. The chapter ends by surveying existing work in the generation of referring expressions. The principal aims of this chapter are to (a) introduce the relevant terminology and (b) provide a characterisation of the subject matter from which we can then focus on the particular areas of interest.

The original material in the thesis begins in chapter 3, where we we turn our attention to the other end of the reference relation, and ask: what kinds of things are referred to? This chapter describes the ontology adopted in the thesis. The basic ideas used here have similarities to the approaches proposed by Link [1983] and Bach [1986]. The chapter specifies the representational formalism used to describe the objects in the universe of discourse. We permit two basic kinds of entities: GENERALIZED PHYSICAL OBJECTS and EVENTUALITIES. The chapter works out the consequences of using these notions, and also points out some of the limitations.

Chapters 4 and 5 describe the approach we take to natural language generation. In chapter 4, the generation framework adopted in this thesis is presented. We describe the mechanisms used to construct a discourse plan, and the construction and use of a discourse model. We also introduce the two intermediate levels of representation used within the system.

In chapter 5, we examine the generation of referring expressions in detail. First, we describe the interface used in EPICURE to mediate between the semantics of a noun phrase and its syntax. We then go on to describe the processes used in EPICURE to determine the content of an initial reference to an entity. Next we consider the problem of deciding when it is safe to generate a pronominal reference to an entity. We then turn to definite noun phrase anaphora, and introduce the notion of DISCRIMINATORY POWER as a tool for determining the required descriptive content of a referring expression. The chapter ends by describing the algorithms required for the generation of *one*-anaphora.

Finally, chapter 6 presents a detailed worked example that demonstrates the mechanisms described in the previous chapters; and chapter 7 summarises the limitations of the present work, offers some conclusions, and suggests some possibilities for future work.

Chapter 2

Reference and Anaphora

This chapter examines the notion of reference, and surveys the variety of anaphoric expressions available in natural language. It then goes on to survey existing work in the generation of referring expressions, before summarising the issues to be addressed in the rest of the thesis.

In section 2.1, we first define some basic terminology and present some of the assumptions underlying the view this thesis takes of the generation of referring expressions. The section includes some discussion of the nature of REFERENCE: what does it mean to REFER to something, and what is it that is referred to? This is not a trivial problem: it has occupied the minds of philosophers for centuries. Some working hypotheses are adopted, the most fundamental being the widely-accepted view that reference is a speech act; we also sidestep some difficult philosophical issues by talking not of REFERENCE but of SPECIFICATION. There follows some discussion of the different kinds of referring expressions available in natural language, and also a brief description of the various non-referring uses of noun phrases.

Sections 2.2, 2.3, 2.4 and 2.5 focus on ANAPHORA. We adopt the definition of anaphora suggested by Hirst:

> ANAPHORA is the device of making in discourse an abbreviated reference to some entity (or entities) in the expectation that the perceiver of the discourse will be able to disabbreviate the reference and thereby determine the identity of the entity. [Hirst 1981a:4]

Each of the different kinds of anaphora we examine has a different focus of interest. In section 2.2, we consider pronominal anaphora in detail, focusing particularly on the variations in the relative *locations* of an anaphor and its antecedent, the variation in the *nature* of the referent of the anaphor, and the variation in the range of relationships that can hold between the anaphor and its antecedent.[1]

In section 2.3, we look briefly at the various forms of definite noun phrase anaphora that are to be found. Our primary interest here is in situations in which it is possible to use a definite noun phrase when referring to an entity: we review various attempts to explain definiteness.

In section 2.4, we look at *one*-anaphora, and present some of the different views taken with respect to the syntactic structure of *one*-anaphoric expressions. The major interest here is the question of what it is that a *one*-anaphor substitutes for.

In section 2.5, we survey previous work in the computer generation of anaphoric referring expressions: three existing systems (Davey's PROTEUS [Davey 1978], McDonald's MUMBLE [McDonald 1980a], and Appelt's KAMP [Appelt 1982, 1986]) are discussed in considerable detail, and a number of other systems are mentioned briefly.

Finally, in section 2.6, against the background of the earlier review material, we present a summary of the questions that can be raised in the context of referring expression generation, and identify those respects in which current systems are deficient. We then specify more precisely those phenomena to be considered in the rest of the thesis.

2.1 Reference

Traditionally, reference is taken to be the relation that holds between a symbol and whatever is signified by that symbol. This view immediately gives rise to two questions: what range of symbols can be used to refer, and what range of things can be referred to? In this thesis, since we are concerned with natural language, we will only consider

[1]The availability of Hirst's extensive survey of anaphora [Hirst 1981a] all but removes the point of providing a survey here; however, I provide an overview that identifies those aspects of particular interest in the context of this thesis. Other interesting and useful surveys of anaphora can be found in Halliday and Hasan [1976], Webber [1979, Chapter 2] and Carter [1987, Chapter 2]; a compressed version of Hirst's survey is available as [Hirst 1981b]. Readers familiar with Hirst's work will recognise some of the examples used in this chapter as being derived from his.

those particular *linguistic* symbols called REFERRING EXPRESSIONS. In this section, we survey a number of different distinctions that are to be found in the literature on reference and referring expressions. We begin by first discussing the notion of reference itself.

2.1.1 What is Reference?

The study of reference has been plagued by subtle terminological differences. For the Greek philosophers, the semantic relationship between words and the corresponding entities in the world was one of NAMING; thus, meaning and reference were equated. In more modern times, this view—known as EXTENSIONALISM, since the meaning of a term is defined as the extension of that term—has been put forward by Russell [1902], and, more recently, by Quine [1960] and Davidson [1967]. The paradigm case of the relationship in question is to be found in the relationship between proper names and the individuals they pick out; thus, the *meaning* of the proper name *Steve Martin* is that well-known comedian. Under this view, common nouns and adjectives are then taken to refer to the sets of individuals of which the corresponding properties can be predicated.

Sense and Reference

As traditional grammar developed, it was realized that the dyadic analysis offered by extensionalism failed to account for certain phenomena in language. To take a well-rehearsed example: if the terms *The Morning Star* and *The Evening Star* both mean the same thing, how is it that the sentence in (2-1) can be informative?

(2-1) The Morning Star is the Evening Star.

Faced with this problem, Frege [1892] distinguished SENSE and REFERENCE. While the reference of an expression is that thing picked out by the expression, the sense of an expression is the means by which the referent is reached, or the 'mode of presentation' of the referent. A number of other essentially similar contrasts are to be found in the literature: in modern formal semantics, the standard terms are INTENSION and EXTENSION.

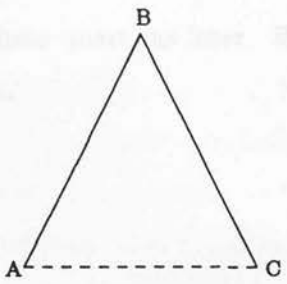


Figure 2.1: The Triangle of Signification

A related, but not identical, triadic analysis of meaning was first introduced by the medieval grammarians, who replaced the simple distinction between sign and signified by a triad in which the form of a word signified a thing in the world by virtue of the CONCEPT associated with the word [Lyons 1968:404]. This analysis was illustrated by Ogden and Richards [1949:11], and has been by many others since, by means of the TRIANGLE OF SIGNIFICATION or SEMIOTIC TRIANGLE, a variation upon which is shown in figure 2.1. Different writers have used different names for the elements and relations in this diagram. For Ogdens and Richards, A is a SYMBOL, B a THOUGHT or REFERENCE, and C a REFERENT. A symbol is said to SYMBOLISE a thought or reference, and a thought or reference is said to REFER TO a referent. The relationships of symbolising and referring to are held to be causal relations, which give rise to a third, indirect, relation (as indicated by the broken line) whereby a symbol is said to STAND FOR a referent. For Lyons [1977:96–97], on the other hand, A is a SIGN, B a CONCEPT, and C a SIGNIFICATUM; in opposition to Ogden and Richard's terminology, Lyons calls the relationship between A and C one of REFERENCE. We will adopt Lyons' terminology, since it is more in agreement with current usage.

This still leaves a number of questions about the status of each of the components of the triad. It is not clear, for example, what we should take the ontological status of the sign and the concept to be: are they physical entities or mental entities? Similarly, we can ask whether the significatum is something referred to on a particular occasion

of use of the sign, or whether it is the totality of things that can be referred to by the sign.

We will return to the first of these questions later. We can deal with the second by introducing a further distinction.

Reference and Denotation

It is useful to draw a distinction between the terms REFERENCE and DENOTATION (or REFERENTIAL RANGE [Lyons 1981:220]). The denotation of a word or expression is the set of things which it denotes, whereas reference is an utterance-dependent notion, relating an expression and what it stands for on a particular occasion of its use. Note that whereas words can have denotations, only complete expressions (which may consist of single words) can refer. Thus, the term *white* DENOTES the class of white things, but does not REFER to anything; the expression

(2-2) the white swan

on the other hand, denotes the set of white swans, but its referent will be a particular white swan whose identity depends upon the context in which the expression is uttered.

What Kinds of Things can be Referred To?

There is also the question of what kinds of things can be referred *to*. In the first instance, we might take physical existence to be a pre-requisite for being a referent. Crystal [1985:259], for example, suggests that the referents of referring expressions are entities in the external world, where by entities he means objects and states of affairs (whatever those are). Lyons [1968:424] suggests that words can stand for 'things, events, actions and qualities'; and Searle leaves things as wide as possible by saying that a referring expression is any expression

> which serves to identify any thing, process, event, action or any other kind of individual or particular
>
> [Searle 1969:27]

Many writers are then willing to extend this basic notion of a referent to cover various kinds of entities that do not have physical existence, such as abstract entities, hypothetical entities and fictional entities.

Although we spoke above of words as signs, it is the case that, in the literature, noun phrases are the only kind of linguistic expressions that are considered as referring expressions.[2] Although some of the possibilities listed above for significata allow that referring expressions may refer to actions, this is not normally intended to imply that verbs, for example, can be referring expressions: rather, the inclusion of actions, processes and the like is primarily intended to cover the use of nominalized verb phrases, such as *his doing of the dirty deed* in example (2-3):

(2-3) His doing of the dirty deed was despicable

In the rest of this thesis, we will adopt this usage, and use the term REFERRING EXPRESSION as a short-hand for NOUN PHRASE REFERRING EXPRESSION.

2.1.2 Kinds of Referring Expressions

There are a number of different ways in which noun phrase referring expressions might be subcategorised. One important subcategorisation distinguishes three kinds of reference: DEFINITE REFERENCE, INDEFINITE REFERENCE, and GENERIC REFERENCE. Each of these is described below, and then a number of other distinctions to be found in the literature are briefly surveyed.

Definite Reference

There are four different kinds of definite referring expressions in English: definite noun phrases, demonstrative terms, proper names and pronouns. These are exemplified in (2-4a) through (2-4d) respectively:

[2] This is not the same as saying that noun phrases are the only expressions that have referents. Thus, following Frege, a sentence has as its referent a truth value; but sentences are not normally talked of as being referring expressions under this view.

(2-4) a the woman with the Victorian values
b that woman over there
c Margaret Thatcher
d she

Definite noun phrases are also referred to (after Russell) as DEFINITE DESCRIPTIONS, thus emphasising that such a referring expression, used in a particular context, can be used to refer by virtue of the fact that it provides a description of the thing that it is referring to. As will be discussed below, not every occurrence of a definite description is a case of definite reference: definite noun phrases can have other, non-referring, uses. Pronouns can also be used for purposes other than reference, although it is less clear whether we can have non-referring uses of proper names or demonstratives.

Singular definite referring expressions are usually taken as the paradigm type of referring expressions, and are the most frequently discussed in the literature. Other types of referring expressions are viewed as being more problematic in one way or another, and rarely discussed.

Indefinite Reference

An indefinite noun phrase is either an indefinite pronoun (such as *one* or *some*) or a noun phrase introduced by an indefinite article. For Russell [1919, Chapter 16], indefinite noun phrases do not refer at all: rather, they serve as existential quantifiers. According to this view, in the sentence

(2-5) A cat rushed through the door.

the indefinite noun phrase *a cat* does not refer to a particular cat; instead, the sentence merely states that the set of entities which are cats and which have at some time rushed through a particular door is not empty.

However, if we do not take *a cat* to be a referring expression, then there is a problem in explaining the function of the pronoun *he* in (2-6):

(2-6) a A cat rushed through the door.
b He looked very hungry.

We stated earlier that a pronoun and its antecedent noun phrase co-refer, but this is clearly impossible if the antecedent noun phrase does not itself refer. We will speak for the moment of indefinite referring expressions, and return later to the question of how it is that they refer.

It is also important to note that indefinite noun phrases can be used in either a SPECIFIC or a NON-SPECIFIC sense. As an example, take an utterance like:

(2-7) Every evening a heron flies overhead.

Under one interpretation, *a heron* here refers to a specific individual: i.e., the same heron every day. A first-order logical form of this instance of SPECIFIC INDEFINITE REFERENCE is as shown in (2-8):

(2-8) $\exists y$ heron$(y) \wedge [\forall x$ evening$(x) \supset$ flies-overhead-at$(y, x)]$

An alternative interpretation takes *a heron* to mean some heron or other: i.e., not necessarily the same one each day. Under this interpretation, the noun phrase *a heron* is being used non-specifically; the sentence then has the following logical form:

(2-9) $\forall x$ evening$(x) \supset [\exists y$ heron$(y) \wedge$ flies-overhead-at$(y, x)]$

Under the second interpretation, it is not clear in what sense the expression *a heron* is being used to refer, since the noun phrase itself does not seem to pick out a single individual. However, just as in the simpler case discussed above, note that subsequent reference can be made to its 'referent': following (2-7,) we could go on to say

(2-10) It usually swoops low over the garden.

This is generally considered to be an instance of BOUND VARIABLE PRONOMINALIZATION (see section 2.2).

Generic Reference

Generic reference takes various forms, and is used in generic propositions such as the following:

(2-11) a *The lion* is a friendly beast.
b *A lion* is a friendly beast.
c *Lions* are friendly beasts.

Here, reference is made, not to an *individual*, but to what we might call a KIND. Note that, at least in the first two cases above, it is not possible to determine from the form of the expression alone whether a noun phrase is being used to perform generic reference: contextual information is required.

The Predicative Use of Noun Phrases

Ignoring the question of whether or not indefinite noun phrases refer, we can ask whether definite noun phrases *always* refer. Evidence that this is not the case is easy to find. For example, the same noun phrase can be used in one situation to refer to an entity, and in another to predicate something of an entity: consider the following examples.

(2-12) a *The fattest man in town* came here yesterday.
b My friend Rupert is *the fattest man in town.*

In the first example, the expression *the fattest man in town* is used to refer to an individual; whereas in (2-12b) the noun phrase *the fattest man in town* is used to predicate something of the individual referred to by the expression *my friend Rupert.*

The Referential/Attributive Distinction

From these examples, we might be led to hypothesise that, whereas noun phrases in object positions may be either referential or predicative, subject noun phrases are necessarily referential. However, even this is not so. Donnellan [1966] distinguishes what he calls the REFERENTIAL and ATTRIBUTIVE uses of definite noun phrases: example (2-13) involves a definite description, which, in the appropriate circumstances, would be an instance of the attributive use.

(2-13) *The person who killed Freddy Frog* is insane.

The idea behind the attributive use of a description is that it is the property of being so-and-so that is most important, irrespective of who or what it is that the expression

picks out in the world. Attributive uses of noun phrases can be distinguished by their ability to have 'whoever it is' appended parenthetically without destroying the meaning of the utterance.

Charniak [1976] and Kronfeld [1986] both suggest computational approaches to the referential/attributive distinction.

2.1.3 Reference as a Speech Act

In the above, we have repeatedly labelled linguistic forms as referring expressions. However, as noted by Strawson [1950] to call any linguistic form a referring expression is misleading, since it is not expressions that refer at all, except in a derivative sense: it is the speaker or writer who refers, by means of expressions of the language. To quote Searle:

> The term 'referring expression' is not meant to imply that expressions refer. On the contrary ... reference is a speech act, and speech acts are performed by speakers in uttering words, not by words.
>
> [Searle 1969:28]

This thesis adopts the speech act approach to reference; however, for brevity and convenience, we will often talk of referring expressions as if the linguistic forms themselves had the property of reference.

In standard speech act theory, as expounded by Searle and others, the basic unit of human linguistic communication is the ILLOCUTIONARY ACT [Searle 1976:1]. Under this view, language use consists in performing illocutionary acts such as making statements, giving commands, asking questions, making promises and so on. In Searle's view, illocutionary acts are only one kind of speech act, however: there are also PROPOSITIONAL SPEECH ACTS, the speech acts of reference and predication. Illocutionary acts are realized by means of propositional acts. I might choose to refer to someone I know as *Steven*, and say of him that he is the fastest typist in the class, by uttering the sentence

(2-14) Steven is the fastest typist in the class.

This single illocutionary act of making a statement consists of two propositional acts: I have made an act of reference using the expression *Steven*, and by means of an act of predication I have predicated of him a certain property.

More recently, Cohen [1981, 1984a, 1984b] has argued that the act of reference is best viewed as a fully-fledged illocutionary act, rather than as a propositional act.

2.1.4 Reference and Specification

Searle states that

> It is characteristic of [singular definite referring expressions] that their utterance serves to pick out or identify one 'object' or 'entity' or 'particular' apart from other objects ...
>
> [Searle 1969:26]

However, exactly what it means to *identify* an entity is unclear, and seems to depend very much on the context. Suppose I say to you:

(2-15) Please bring me the red block.

Here it is obvious that, in order for you to satisfy the request, you have to identify the real-world entity described by the expression *the red block*, where identification means actually being able to locate the entity in the world. A weaker notion of identification seems to be required in the following sentence:

(2-16) Julius Caesar never shopped at Safeway.

Here, it is unlikely that you could ever identify the referent of *Julius Caesar* in the same sense of identification as required in the previous example; and yet, we would want to say that *Julius Caesar* is a referring expression. Also, as we saw earlier, we may refer to entities which do not exist in the actual world: we can ask, then, what sense of identification is required to explain references to fictional entities, as in

(2-17) Captain Kirk bought his favourite kind of oatcakes.

To get around this problem, Lyons suggests the following:

> Once any information at all has been supplied about an indefinite referent, it can then be treated by the participants as an individual that is known to both of them and identifiable within the universe of discourse by means of a definite referring expression. It is not a necessary condition of successful reference that the speaker or hearer should be able to identify the individual being referred to in any sense of 'identification' other than this.
>
> [Lyons 1977:189]

To deal with this in a computational framework, following Sidner [1979] we replace the notion of reference by a notion of SPECIFICATION; referring expressions are then used to specify objects in the universe of discourse, and not objects in the world. Thus, when a speaker uses a referring expression, he or she is referring to an object in his or her discourse model, with the intent of causing the object in question to be introduced or identified in the hearer's discourse model. This approach has been adopted either implicitly or explicitly in a great deal of work in computational linguistics; the notion of a DISCOURSE REFERENT was first suggested by Karttunen [1976], and has subsequently been adopted by a great many researchers under sometimes different names. Thus, Webber [1983], and many others, talk of DISCOURSE ENTITIES; and, in work with a more psychological orientation, Johnson-Laird and Garnham [1980] talk of referring expressions picking out entities in a DISCOURSE MODEL. We will return to consider the notions of discourse entity and discourse model in more detail in chapters 4 and 5.

Strictly speaking, adopting this approach requires some changes of terminology. Previously, we talked of an anaphor and its antecedent *co-referring*. Following Sidner, we should now talk of an anaphor and its antecedent CO-SPECIFYING. That is, instead of talking about the relationship of reference—the relationship between a linguistic expression and that entity in the world which it picks out—we should talk about the relationship of SPECIFICATION: the relationship that holds between a linguistic expression and that element in the universe of discourse (or DISCOURSE ENTITY) that it specifies. The discourse entity then *represents* the real world entity in question. The relationship of reference is therefore replaced by a pair of relations, those of specification and representation. As an alternative terminology, Webber [1988] suggests the terms REFER$_m$, meaning 'refer in a model', and REFERENT$_m$, for the entity in the model picked out by the referring expression. By focusing our attention on specification, or reference$_m$, we can avoid many of the philosophical questions concerned with how an expression relates to the real world. In particular, it allows us to treat definite and indefinite reference

more uniformly: broadly speaking, indefinite reference introduces an entity into the discourse, and definite reference picks out an entity already in the discourse.

To ease the flow of exposition, we will continue to talk of *reference* in the present work. The reader should bear in mind, however, that the intended notion is really that of specification.

2.2 Pronominal Anaphora

The most widely discussed anaphoric form is the pronoun; and of the various pronominal forms available, it is the third person singular that has been examined in most detail. Even here, the genderless form *it* has been scrutinised more closely than the masculine and feminine forms *he* and *she*. A great deal of research in the linguistics literature focuses on the behaviour of anaphoric relations within individual sentences (INTRA-SENTENTIAL ANAPHORA): the concerns here are with elucidating the syntactic constraints obeyed by pronominal anaphora. In the computational linguistics literature, more effort has been expended on examining the relationships between pronouns and their antecedents in earlier sentences in connected discourse (INTERSENTENTIAL ANAPHORA). It seems to be generally assumed that restrictions on pronominalization apply equally to each of the third person forms, although the wider variety of uses to which the genderless form is put might cause us to question this: for example, as well as being used to refer to genderless entities, the word *it* can be used in situations where no analogous use of *he*, *she* or any of the plural forms is possible:

(2-18) a *It* is raining.
b *It* is because of this that ...

In the linguistics literature, an important distinction is drawn between BOUND VARIABLE PRONOMINALIZATION [Partee 1978] and PRAGMATIC or REFERENTIAL pronominalization.[3] Bound variable anaphora can be interpreted purely at the level of semantics, without any need to identify a referent in the world: in the logical form representation of sentences containing bound variable anaphora, the anaphor and its antecedent are represented by the same variable. Thus, the sentence

(2-19) Every cat who finds *a blanket* sits on it.

[3]Bosch [1983] refers to bound variable pronouns as SYNTACTIC pronouns.

has the logical form

(2-20) $\forall x, y\ \text{cat}(x) \wedge \text{blanket}(y) \wedge \text{finds}(x, y) \supset \text{sits-on}(x, y)$

In the case of pragmatic pronominalization, however, pragmatics is required in order to determine the referent of a pronoun, which is some entity in the world, as in the following example:

(2-21) a William tried to catch a spider.
b It was too fast for him.

In this thesis, our primary interest is in pragmatic anaphora, and so we will have nothing more to say about bound variable anaphora.

Different uses of the pronominal form can be distinguished in a number of different ways:

- in terms of the relative locations of the pronoun and its antecedent;
- in terms of the kinds of entities which can serve as the referent of a pronoun; and
- in terms of the semantic relationship between the referent of the pronoun and its antecedent.

We will consider each of these in turn.

2.2.1 The Location of the Antecedent

A pronoun may refer to an entity mentioned in the same sentence or an earlier sentence; if the antecedent noun phrase is in the same sentence, it may occur in a variety of locations relative to the anaphor, both preceding and following it. If the antecedent is in an earlier sentence, it is generally to be found in the immediately preceding sentence, although there are exceptions to this.

Pronominal Reference to Entities in the Same Sentence

A pronoun may have as its antecedent a noun phrase that appears earlier in the same sentence. In such cases, the antecedent may be found in a separate clause, as in (2-22a),

or in the same clause, as in (2-22b):

(2-22) a Because Judy was passing the pet shop, she was asked to buy a ten kilo bag of cat litter.

b Mary brought a gallon of chilli sauce and a container to store it in.

Note that pronominal reference is also possible from within a relative clause to the head of that relative clause:

(2-23) a block which is bigger than anything which supports it

Within linguistics, there has been substantial interest in the configurational restrictions on intra-sentential pronominal reference. In (2-24a), *he* and *William* cannot co-refer, whereas in (2-24b) they can:

(2-24) a He loves William's new box.

b William loves *his* new box.

Lasnik [1976] and Reinhart [1976, 1981, 1983] offer rules that attempt to specify the relevant syntactic constraints. There are also constraints governing the use of the reflexive pronominal form as in:

(2-25) The cat put itself among the pigeons.

Popowich [1988] examines reflexives in detail. Since our primary interest is in intersentential anaphora, we will not pursue the issues here any further.

Cataphora

It is not necessary for a pronoun to follow its antecedent: the pronoun can appear prior to the full noun phrase with which it co-refers:[4]

(2-26) Because she was passing the pet shop, Judy was asked to buy a ten kilo bag of cat litter.

Again, a considerable amount of interest within linguistics has focused on the constraints on this use of pronouns.

[4] We ignore here the use of 'long-distance cataphora' as a device for catching the reader's interest, as is found in novels that begin with sentences like *He opened the door carefully ...*

Pronominal Reference to Entities in the Preceding Discourse

A great amount of work in natural language understanding has focused on the problem of developing algorithms for determining the antecedents of pronouns that appear in the preceding discourse. The most well-known work in this area is that of Sidner [1979] (described later) and Webber [1979]; Carter [1987] describes a recent extension of Sidner's work.

In most instances, the antecedent of a pronoun is to be found in the preceding sentence, thus obeying what Pinkal [1986:370] has called a LOCALITY CONSTRAINT. In a relatively small number of cases, the antecedent of a pronoun may be found some way back in a text. Hirst [1981b] offers this example:

(2-27) Just as Carrie, played by Sissy Spacek, can be seen as another of De Palma's ambiguous women, as in Obsession, other parallels in the construction of the two films rapidly spring to mind. One can compare, for example, the extraordinary power of the final moments of the present film, in which the gentle, sunlit, Vaseline-lensed scene is shattered by a sudden horror that makes many people literally jump out of their seats, with that of Obsession, wherein the unexpected again happens, though this time in the negative sense that the expected does not happen.

However, despite De Palma's skill, it is her acting that ultimately makes the film.

Here, the anaphor *her* in the last sentence refers back to *Sissy Spacek*, mentioned at the beginning of the preceding paragraph.[5] Such uses of the pronominal form are relatively rare, and it is often not clear whether they are legitimate: we will discuss this issue further in chapter 5.

Exophoric Reference and Deixis

Sometimes a pronoun does not have an antecedent in the text; instead, the referent is available directly in the external world. Thus, we find deictic uses of the pronominal form in sentences like the following, uttered in the appropriate contexts (e.g., where both the speaker and the hearer are focusing their attention on some visible object):

[5] *De Palma* cannot be the antecedent since the referent—Brian De Palma—is male.

(2-28) a Pick that up and put it over there.
b Do you like it? Terry bought it at Dixon's.

Such uses of the pronominal form are beyond the scope of this thesis.

2.2.2 The Nature of the Referent

The entity picked out by a pronoun can be of several different types. More often than not, the referent of an anaphor is an individual explicitly referred to previously in the text.

(2-29) a *The cat* is out.
b Who will let him in?

However, anaphoric reference can also be made to events. Sometimes the event in question will have been previously referred to by means of a nominalization:

(2-30) a Vina asked Jon to come to *the opening of the exhibition.*
b It was going to be a posh affair.

Alternatively, the antecedent may be a complete clause or sentence, as in (2-31):

(2-31) a Ewan was buttering the toast while feeding Tom and Eleanor with the other hand.
b It saves time.

Here the referent of the pronoun is the entire event described in the previous clause.

2.2.3 The Semantic Relation of the Referents

In the previous examples, the relation between the anaphor and the antecedent consisted in their identity of reference. However, there are also more complex relations that can hold between an anaphor and its antecedent.

First, a pronoun may refer to an implicitly constructed set, as in (2-32):

(2-32) When Joe went to see his solicitor, they spent a rather long time rewriting the contract. When Dom eventually arrived, Joe fooled his solicitor into leaving the room, whereupon he and Dom slipped out the back door. They laughed all the way to the bank.

The first occurrence of *they* here refers to a set consisting of Joe and his solicitor, whereas the second occurrence refers to a set consisting of Joe and Dom.

It need not even be the case that the antecedent is mentioned explicitly in the text. Hirst [1981b] offers the following example:

(2-33) a Ross sat in the corner, knitting madly. Suddenly he threw it down, and stormed out of the room.

b Ross wanted to NAIL the boards together, but Sue made him do it with TAPE.

There are also cases of what Hirst [1981b] has called STRAINED ANAPHORA:

(2-34) John became a guitarist because he thought that it was a beautiful instrument.

Such instances are relatively rare, and will not be considered in the present work.

Finally, as we noted earlier, not all instances of pronouns are anaphoric.

(2-35) It is raining.

It might be suggested that the *it* here is being used to refer to something like 'the general situation', but this seems implausible.

2.2.4 When Can a Pronoun Be Used?

The problem of pronominal reference resolution has loomed large in work on natural language interpretation. The most well known work in this area is that of Sidner [1979], who developed a set of pronoun interpretation rules (discussed further below): the emphasis here was on the development of an algorithm which would select the most likely candidate referent for a particular pronoun. Addressing another aspect of the same problem, Webber [1979] focused on a formal treatment of what entities were made available for anaphoric reference within a discourse.[6] More recently, Carter [1987] has extended Sidner's algorithms and integrated them with Wilks' PREFERENCE SEMANTICS,[7] resulting in a system which requires only a limited amount of world knowledge in order to resolve

[6] Thus, whereas Sidner's work was concerned with choosing the most likely candidate for an antecedent, Webber's work was concerned with establishing exactly what the set of candidates might be. Unfortunately, no one appears to have attempted to integrate the two lines of work.

[7] See, for example, Wilks [1975].

pronominal references; and Grosz, Joshi and Weinstein [1983] have suggested a notion of CENTERING as a simpler alternative to Sidner's work.

We briefly overview Sidner's approach below, before considering Grosz, Joshi and Weinstein's [1983] notion of centering.

Sidner's Focusing Algorithms

Sidner [1979, 1981] describes a mechanism for interpreting pronouns using a notion of FOCUS and a set of pronoun interpretation rules, called PI-RULES. A principal theme of Sidner's work is that any theory of anaphora interpretation must take into account the fact that various factors, syntactic, semantic and pragmatic, play a role in anaphora interpretation. The need for pragmatic information in pronoun resolution is evidenced by examples like the following:

(2-36) a The councillors refused the women a permit because they condemned revolution.

b The councillors refused the women a permit because they supported revolution.

Note that selectional restrictions are an inadequate means of choosing the correct antecedents in cases like these: in (2-36a), for example, the preferred reading is that the pronoun co-refers with *the councillors*, but one can imagine an editorial in the *Sun* berating a left-wing council, and intending the pronoun to co-refer with *the women*.

In Sidner's theory, the role of inference in anaphora resolution is changed from that of blind search into a constraint-checking process: syntactic and semantic information is used to order the candidate antecedents, and an inference mechanism is then used to check each proposed antecedent in turn. The algorithms involved are quite complex, involving a substantial number of rules. The notion of focus plays a central role here: each utterance has a POTENTIAL FOCUS LIST (the entities mentioned in the sentence), and both an ACTOR FOCUS and a DISCOURSE FOCUS. The rules specify the order in which candidate antecedents are checked, and specify the conditions under which the foci may change.

The role of focus in language comprehension seems relatively straightforward. The

hearer settles on some entity that he or she thinks is the focus of the discourse, and uses this supposition to interpret anaphoric expressions. The hearer also takes account of shifts in focus, and can reject suppositions where necessary, finding an alternative candidate for the referent of an anaphoric expression.

Some researchers working in natural language generation have suggested that Sidner's rules—or more precisely, a notion of focus akin to that used by Sidner–are also relevant from the point of view of natural language generation. McDonald [1980:220] suggests that focus might be sufficient for determining whether pronominalization is possible; Appelt [1982:129–130] and McKeown [1982:124–132] go further, and describe how the interpretation rules might actually be adapted for use in generation. Ultimately, the most interesting use is that made by McKeown, who suggests that the rules cannot just be inverted for use in generation: rather than simply use focus as a means to determine how to refer to entities (i.e., to decide when a pronoun can be used to refer to an entity given that the decision has been made to refer to that entity), she also uses the notion of focus to decide *what* to say: it is then a desire to maintain, rather than shift, the focus of attention that determines which propositions are realized. This seems a more plausible use of the notion of focus, since, of course, it is the speaker who decides what she is going to focus her attention on.

Grosz [1977] hypothesizes a distinction between LOCAL FOCUS (or IMMEDIATE FOCUS) and GLOBAL FOCUS. Sidner's algorithms were an attempt to flesh out the notion of local focus, complementing Grosz's work on global focus. However, the mechanisms she eventually proposed essentially ignored the notion of global focus, and, in extreme cases, would search a FOCUS STACK for candidate antecedents in situations where we might have expected discourse structure to be considered. In the next section, we briefly review an approach to pronoun resolution that is better integrated with the view of discourse structure described later in the present chapter, and adopted in the approach we take to discourse modelling in chapter 4.

Grosz, Joshi and Weinstein's 'Centering'

As a simpler alternative to Sidner's proposal, Grosz, Joshi and Weinstein [1983] suggest a theory of pronominalization based on a notion of CENTERING. This simplicity derives

from two factors: first, the theory appears to be less developed (and so its simplicity might diminish as it is developed further); and secondly, it is essentially a speaker-based approach.

Under this theory, every sentence has a number of FORWARD-LOOKING CENTERS[8] (C_f) and a single BACKWARD-LOOKING CENTER (C_b). The forward-looking centers correspond to Sidner's potential focus list; the backward-looking center, usually called just the CENTER, is essentially the sentential topic, i.e., what the sentence is about, and thus corresponds to Sidner's discourse focus. Grosz *et al* suggest a rule for pronominalization as follows:

(2-37) a pronoun can be used to refer to an entity provided either *(a)* the entity to be referred to is the center of the preceding utterance, or *(b)* if it is not the center of the preceding utterance, then any reference to the center is also pronominalized.

This rule does not prohibit the use of pronouns to refer to any other entities mentioned in the preceding sentence, provided the center, if it is maintained, is referred to by means of a pronoun. The motivation behind this restriction is that if the center of the preceding utterance is not referred to in the current sentence by means of a pronoun, but instead by means of some other form of reference, and if some other entity in the previous utterance is referred to by means of a pronoun, then this is potentially confusing for the hearer, since the pronominal form is expected as the means of reference to the center.

This proposal seems more appropriate in the context of language generation; however, it again suffers from the problem that it relies on an intuitive notion, that of the center. No rules are specified for determining what the center of a given sentence is. In chapters 4 and 5, we introduce a related notion (called the CENTRE of an utterance) that is sufficiently well-defined to be useful in the particular domain of application of the system described in this thesis, but is less general than that which Grosz, Joshi and Weinstein have in mind.

[8]Since the term is used in a technical sense, we use the American spelling of the original authors here.

2.3 Definite Noun Phrase Anaphora

Traditionally, it is considered to be the case that entities are introduced into a discourse by means of indefinite reference, whereas subsequent references to already introduced entities are usually of the form of definite referring expressions. Thus, we have the following discourse:

(2-38) a *A man* and *a woman* walked into the room.
b The man was wearing a funny hat.

Unfortunately, the matter is not that simple, since we can easily find cases where the initial reference to an entity within a discourse contravenes this rule:

(2-39) a A bus turned the corner.
b The driver had a mean look in his eye.

Here, a definite noun phrase (*the driver*) is used to refer to an entity the first time it is mentioned. Of course, this is a widely recognized phenomenon: some referring expressions refer to entities in such a way that it seems the entities in question are assumed known to the hearer, while others suggest that the entities referred to are new to the hearer. One set of terminology that has been developed for this dichotomy in information status is that of the pair of concepts GIVEN and NEW. Characteristically, new information is introduced by indefinite expressions; thereafter, any entity so introduced is considered given, and subsequently referred to by means of a definite expression. However, as we have just seen, some entities appear to be given without first being explicitly introduced into the discourse; thus, a more satisfactory account of the uses of definite and indefinite reference is required. Before we go on to consider the work of various researchers who have attempted to provide such an account, we will first look briefly at some variations in the more straightforward use of the definite determiner.

2.3.1 Referring to Already Mentioned Entities

There are a number of different ways in which a definite noun phrase can be used to refer to an entity that has already been explicitly introduced into the discourse.

The most common form of definite noun phrase reference is the PARTIALLY REPEATED

form, where some part of the noun phrase used to introduce an entity into a discourse is used to refer to it subsequently. Typically the repeated form consists of the head noun, and as many modifiers as are necessary to distinguish the intended referent from other entities with which it might otherwise be confused. So, in the simplest case, we might have

(2-40) a We have a cat and *a fish*.
b The fish is not particularly friendly.

In a slightly more complicated situation, we might have

(2-41) a *The large green ball* is between the small blue ball and the large red ball.
b Pick up the green ball.

Here, we have to specify a property of the intended referent, in addition to the fact that it is a ball, in order to distinguish it from the other entities that have been mentioned.

A less common use of the definite noun phrase is to refer back to an already introduced entity using a different lexical head, as in the following examples:

(2-42) a *Bunny's son* is ill again.
b The child is hardly ever well.

(2-43) a Jeremy used *a lathe* to turn the metal.
b The machine wasn't up to the job.

In such cases, the head of the anaphor is typically a term superordinate to that used in the antecedent: thus, the following is somewhat odd if *a machine* and *the lathe* are supposed to be co-referential:

(2-44) a Jeremy used *a machine* to turn the metal.
b #The lathe wasn't up to the job.

There are various stylistic reasons that can account for the use of lexical replacement. In certain genres, it functions as a means of introducing additional information about the referent:

(2-45) a Today *Big Ben* struck for the last time.
b The 100 year-old clock has slowly disintegrated.

Epithets are also a form of lexical replacement:

(2-46) Tam hates *politicians*, because the bastards always lie.

We will restrict ourselves to the use of the partially repeated form, and some instances of lexical replacement where superordinate terms are used.

2.3.2 Definiteness and Indefiniteness

The determiners *the* and *a* are amongst the most common words in the English language,[9] and there is no shortage of theoretical work which has addressed the questions of their meaning and use. In this section, we review some of the work in this area.

Russell's Logical Analysis of the Determiners

Early work in the philosophy of language attempted to explain the definite and indefinite determiners in terms of logic: thus, for Russell, the sentence

(2-47) The king of France is bald.

was held to be a statement of existential uniqueness, equivalent to the assertion of the conjunction of the following three propositions:

(2-48) a There is a king of France.
b There is not more than one king of France.
c This individual is bald.

The indefinite determiner, under this account, is used to assert existence without asserting uniqueness.

There are various well-known problems with this account as stated. Although it might be quite suitable for determining the logical truth or falsity of a sentence, it has little to offer in terms of an explanation of language use. For example, if I say

(2-49) The cat is in the kitchen.

[9] In a corpus analysed by Kučera and Francis [1967], *the* is the most frequently used of 50406 distinct words: it occurs 69971 times in a corpus of 1,014,232 words (i.e., *the* accounts for 6.9% of all words in the corpus). The word *a* occurs 23237 times, ranking fifth. The ratios for high frequency words are fairly consistent across the different genres Kučera and Francis considered.

it can hardly be claimed that I am asserting that there is only one cat. We might suggest, then, that instead of interpreting *the F* as denoting x if and only if x is the one and only F in existence, we should rather interpret *the F* as denoting x if and only if x is the one and only F in some contextually-defined domain of discourse. The problem is then one of determining what, in any given situation, the required domain of discourse is. We will return to this question below.

It is not only the uniqueness aspect of definite reference that is troublesome. Even the requirement of truth needs to be reconsidered, since an expression which is untrue of an object can still be used to refer successfully to that object: thus, to repeat the oft-cited example, I can successfully identify an individual to you by saying

(2-50) The man holding the martini is Noam Chomsky.

in a situation where it transpires that the man is actually holding a glass of water. Truthfulness is less important than usefulness in identification:

> the more basic and more general notion governing the use of definite descriptions is that the hearer can be assumed to be capable of identifying the referent on the basis of the properties ascribed to it, whether correctly or not, in the description.
>
> [Lyons 1977:182]

Below, we will examine some alternative approaches to definiteness and indefiniteness. We consider a number of ways of determining the relevant CONTEXT within which a definite determiner is licensed. First we look at various approaches to the use of definite determiners in terms of shared knowledge; then we consider the related question of salience.

Searle [1969] attempted to make up for some of the inadequacies in the logical approach to definite referring expressions by redefining their conditions for use in terms of speech act theory. We begin with work by Hawkins, which takes Searle's approach as its starting point.

2.3.3 Shared Knowledge

Hawkin's Location Theory of Reference

Hawkins [1978] provides a more elaborately worked out theory of definiteness and indefiniteness in speech act terms, called the LOCATION THEORY OF REFERENCE. For Hawkins, an entity can be referred to using a definite noun phrase if that entity belongs to what he calls a SHARED SET [Hawkins 1978:130]. He distinguishes a number of such shared sets: the PREVIOUS DISCOURSE SET contains those entities which have already been mentioned in the current discourse, and any other discourses the current speaker and hearer have participated in; the IMMEDIATE SITUATION SET contains those entities which are evoked 'situationally', i.e., by being present in the environment; the LARGER SITUATION SET contains those entities which are given by virtue of being part of specific or general background knowledge; and the ASSOCIATION SET contains those entities whose existence is inferrable on the basis of association with entities which are already known. This allows Hawkins to postulate explanations for many instances of use of the definite determiner, but not for all instances: in particular, Hawkins does not posit a 'shared set' analysis in the case of sentences like the following.

(2-51) a The woman Jo went out with last night hit him in the eye.
b William is amazed by the fact that the moon isn't made of cheese.

For Hawkins, these are instances of the UNAVAILABLE USE of the definite determiner, and in such cases he invokes a transformational analysis. The implication is that the surface structure definite is derived from an indefinite, and so there is no suggestion that the entity in question is mutually known.

Apart from instances of the type just mentioned, Hawkins' claim is that, when a speaker uses a definite referring expression, he or she does the following:

- introduces a referent (or referents) to the hearer;
- instructs the hearer to locate the referent in some shared set of objects;
- refers to the totality of the objects or mass within this set which satisfy the referring expression.

The use of shared sets thus solves the problem of the 'scope of uniqueness' introduced by the earlier logical analysis.

Since Hawkins' approach is based in speech act theory, he goes into considerable detail in characterising the APPROPRIATENESS CONDITIONS for an act of definite reference [Hawkins 1978:167–168]. His analysis provides an important generalisation of previous attempts at speech act analysis of reference. Searle's earlier attempts [Searle 1969] were directed exclusively at singular definite reference: Hawkins' analysis also handles plural definite reference, since he construes the definite article *the* not as indicating uniqueness of referent, but rather as indicating exhaustiveness: when a definite referring expression is used, the speaker intends it to refer to all the objects that the attached description applies to. Thus, if I say

(2-52) Bring in the milk bottles.

I mean *all* the milk bottles. Singular definite reference thus becomes a specific case of this analysis, where the set of objects being referred to contains only one element: uniqueness is a specific type of exhaustiveness.

Viewing the use of the definite article in this way then leads Hawkins to propose a similar speech act analysis for the use of indefinite referring expressions. The fundamental characteristic of indefinite reference is that it is always *exclusive*: if an indefinite referring expression is used, there is always at least one element in the set of potential referents which is being excluded. Thus, it would be inappropriate for me to say

(2-53) Bring me a book.

if there is only one book in the intended shared set. A speaker who uses an indefinite reference then does the following:

- introduces a referent (or referents) to the hearer;
- refers to a proper subset, i.e. not all, of the potential referents of the referring expression.

The indefinite referent may or may not be locatable in a shared set: again, Hawkins provides appropriateness conditions for each case.

Clark and Marshall's Mutual Knowledge

Clark and Marshall [1981] take Hawkin's location theory of reference as a starting point, and develop a theory which tries to overcome its weaknesses. In particular, they note that Hawkins' theory implicitly assumes that the time at which the shared knowledge is acquired must precede the time of the act of reference. For each use of the definite article suggested by Hawkins, Clark and Marshall show that counter-examples to this supposition can be found.[10] So, for example, the utterance

(2-54) Please pass the salt.

can be felicitous even if the hearer is not aware that there is in fact any salt around. They suggest that it is the act of reference itself (in conjunction with mutual knowledge) that causes the hearer to look for a referent. By allowing that the time of acquisition of shared knowledge does not need to precede the act of reference, Clark and Marshall are then able to posit a non-transformational account of the unavailable use examples which were problematic for Hawkins.

For Clark and Marshall, the use of the definite determiner in direct definite referring expressions adheres to what they call the DIRECT DEFINITE REFERENCE CONVENTION [Clark and Marshall 1981:26]: this states that, for a speech act of reference to be successful, the entity referred to must become mutually known to both the speaker and the hearer. If the entity is not already mutually known, then, suggest Clark and Marshall,

> ...to become mutually known, the referent must at least be anchored to something that is already mutually known via an anchor cable that is already mutually known.
>
> [Clark and Marshall 1981:26]

Typically, the anchor is a mutually known entity, and the anchor cable is a mutually known fact about the world. This analysis can then be used to explain an utterance

[10] Clark and Marshall also note another problem with Hawkins' theory: it does not deal with INDIRECT referring expressions such as *the ham sandwich at table twelve* where the intended referent is in fact the person who has ordered a ham sandwich. Clark and Marshall themselves have nothing to say about the question of indirect reference, other than to explicitly restrict their attention to direct definite reference. For an approach to indirect definite reference, see Nunberg's notion of *pragmatic functions* [Nunberg 1978].

like

(2-55) I wonder where the town hall is.

when driving through a city of a certain size for the first time: the town hall does not need to be mutually known, but the anchor (the town in question) is mutually known, and so is the anchor cable (the fact that towns of this size usually have a town hall).

The anchor may be mutually known by virtue of its physical presence, or it may have to be contained within the reference itself to make the reference felicitous, as in the following example:

(2-56) The woman Jo went out with last night hit him in the eye.

where *Jo* is the anchor.

Just as Hawkins made use of several distinct shared sets, Clark and Marshall introduce several basic categories of mutual knowledge.

First, there is mutual knowledge based on COMMUNITY MEMBERSHIP: this is knowledge shared by virtue of belonging to the same communities as the hearer. This type of mutual knowledge is preserved over long periods and used repeatedly.

The other types of mutual knowledge are more temporary, in that they are only relevant for short periods of time.

The strongest possible evidence for mutual knowledge is that based on PHYSICAL COPRESENCE. Clark and Marshall distinguish three types of physical copresence: POTENTIAL, IMMEDIATE and PRIOR. The first of these causes the hearer to search for the object in question; the second holds when both speaker and hearer are focusing upon the entity in question; and the third is used where the entity in question was being looked at prior to the utterance, but not at the actual time of utterance. Typically, this last type of physical copresence would result in the use of the determiner *that*.

The third basic category of mutual knowledge is that based on LINGUISTIC COPRESENCE: entities whose existence (real or imaginary) has been posited linguistically are said to be linguistically copresent. These are entities which have been introduced into

the discourse. PRIOR linguistic copresence is the standard case, where the entity is posited before a reference which assumes mutual knowledge is made; POTENTIAL linguistic copresence is introduced to explain cataphora.

An act of referring may involve a mixture of linguistic and physical copresence and community membership as its basis for mutual knowledge. In

(2-57) I bought a candle yesterday, but the wick had broken off.

we have an example of indirect linguistic copresence: the candle is mutually known because it is linguistically copresent, and because the following generic information is mutually known on the basis of community membership:

(2-58) Candles have wicks.

the candle's wick is also mutually known, and therefore can be referred to by means of a definite referring expression.

Prince's Taxonomy of Given–New Information

In order to clear up the terminological mess created by the differing uses of the term GIVEN, Prince [1981] proposes an alternative terminology. In her framework, a fragment of text or speech is to be viewed as a set of instructions to the hearer to construct a discourse model. Discourse models are said to contain DISCOURSE ENTITIES, ATTRIBUTES and LINKS between entities. Each entity has an INFORMATION STATUS. This is a measure of the assumed familiarity of the entity, and is one of the following: textually evoked, situationally evoked, unused, inferrable, contained inferrable, brand new (anchored), and brand new, as shown in figure 2.2.

UNUSED entities are entities which are assumed by the speaker to be in the hearer's background knowledge, but not in his or her consciousness. i.e., they are mutually known by the speaker and hearer, but have not yet been mentioned in the discourse. Under appropriate conditions of use, each of the following referring expressions can refer to an unused entity:

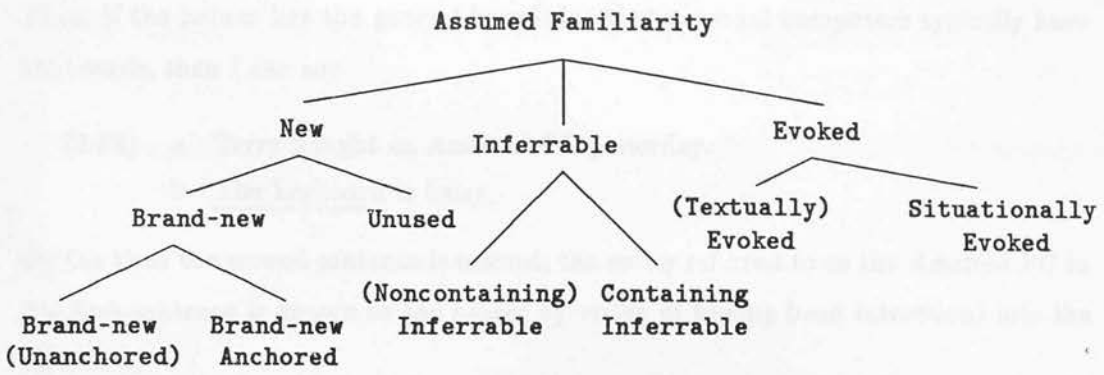


Figure 2.2: Prince's Taxonomy of Given-New Information

(2-59) a Margaret Thatcher
b the moon
c my cat

In Hawkin's [1978] terms, the entity in question is a member of a shared set (other than the previous discourse shared set).

A BRAND-NEW UNANCHORED entity is one which is not already present in the discourse model, and which is not known to the hearer. Such entities are always referred to by means of indefinite noun phrases:

(2-60) I bought a new pen yesterday.

A BRAND-NEW ANCHORED entity is one which is not already in the discourse model, and which is not already known to the hearer, but which, when introduced into the discourse, is related to an entity that *is* known to the hearer. Brand-new anchored entities may be referred to by means of indefinite or definite noun phrases:

(2-61) a a guy I met at the Coarse Revue
b the woman Jo went out with last night

Inferrable entities are those which the speaker assumes the hearer can infer from discourse entities which have already been introduced. A NONCONTAINING INFERRABLE

entity is one which is not present in the discourse model prior to being referred to, but whose existence can be inferred by the hearer on the basis of background knowledge. Thus, if the hearer has the general knowledge that personal computers typically have keyboards, then I can say

(2-62) a Terry bought an Amstrad PC yesterday.
b The keyboard is lousy.

By the time the second sentence is uttered, the entity referred to as the *Amstrad PC* in the first sentence is known to the hearer by virtue of having been introduced into the discourse.

An entity which is a CONTAINING INFERRABLE is one whose existence can be inferred by the hearer by means of a link explicitly supplied by the speaker, as in

(2-63) one of the eggs

A TEXTUALLY EVOKED entity is one which is already present in the discourse model, and is being referred to for the second or subsequent time.[11]

An entity which is SITUATIONALLY EVOKED is one which is mutually known by virtue of its presence in the shared environment of the discourse participants, such as those referred to by expressions such as *I* and *you*.

Prince suggests [1981:245] that these different kinds of assumed familiarity lie on a scale of preference which governs the speaker's choice of form of reference. Where possible, reference to an evoked entity is most preferred, and reference to brand new entities is least preferred:

(2-64) $\begin{matrix} E \\ E^S \end{matrix} > U > I > I^C > BN^A > BN$

Brown and Yule [1983] go on to show that an analysis of a small set of data[12] provides

[11]Brown and Yule [1983:183–185] suggest that different linguistic realisations found for textually evoked entities justifies a further distinction within this category, between current and displaced evoked entities. If we have an expression referring to the most recently introduced entity previous to the newest entity in a discourse, as in *Draw a line, and draw another line across it*, then this expression is said to refer to a current given entity. References to any entities introduced prior to the current given entity are references to displaced given entities. The intuitions underlying this distinction are essentially those that motivate Sidner's [1979] notion of focus shifting, discussed earlier.

[12]It should be noted that, as Brown and Yule themselves point out, the data they examine (namely, descriptions of diagrams) is very restricted. It is not clear how generalisable their conclusions are.

some interesting results:

- expressions of the form *the + property + noun* are used almost exclusively to identify displaced entities;
- the simple definite *the + noun* is used predominantly in identifying displaced entities, but is also used in identifying current entities;
- pronouns never occur as expressions identifying displaced entities, but only as expressions identifying current entities;
- brand new entities are often introduced by *a (+ properties) + noun*;
- inferrable entities are usually introduced by means of definite expressions;
- current textual evoked entities are usually referred to by means of pronominal or elided expressions, although definite referring expressions are also used: the general principle appears to be that the most minimal referring form possible is used, since the entities in question can be assumed by the speaker to be in the forefront of the hearer's mind;
- displaced textual entities, in the data considered by Brown and Yule, are never referred to pronominally or elided, but are always referred to by definite referring expressions, often accompanied by an identifying property.

2.3.4 Salience Rankings

Lewis [1979:239–240] takes the view that it is insufficient to interpret *the F* as denoting x if and only if x is the one and only F in some contextually-defined domain of discourse: for, we can say

(2-65) The dog got in a fight with another dog.

He suggests the following interpretation of *the F*:

> *the F* denotes x if and only if x is the most salient F in the domain of discourse, according to some contextually determined salience ranking
>
> [Lewis 1979:241]

Fleshing out the relevant notion of salience in a form that is computationally useful

is not easy. In this section, we look at a number of discussions of this and related concepts in the literature. Broadly speaking, there are two general approaches to the question of relative salience. One takes the view that each entity in a discourse can be assigned a value on a continuous scale of ACTIVATEDNESS; the entity with the highest activation level is then the most salient. The other view posits a two level distinction between LOCAL FOCUS and GLOBAL FOCUS, analogous to the psychological distinction between short term and long term memory. First we consider the notion of CONCEPT ACTIVATEDNESS suggested by Kantor. The other school is best exemplified by the work of Grosz and Sidner, which we consider in some detail in the following section.

Kantor

Kantor [1977] is concerned with what he calls the CONSIDERATE use of pronouns in discourse, although the general ideas that he discusses are not necessarily restricted to pronominal reference. He introduces a notion of CONCEPT ACTIVATEDNESS, and suggests that before a pronoun can successfully be used to refer to an entity, that entity must have a sufficiently high degree of activatedness in the hearer's mind. The degree of activatedness of a concept is, Kantor suggests, dependent upon factors such as the discourse function of the utterances within which it is referred to, the linguistic construct used, the syntactic position of the antecedent, and what the current topic of the discourse is.

The notion that each entity or concept evoked in a discourse has an associated non-discrete activation level is an intuitively appealing one, perhaps because it has a ring of neurophysiological plausibility about it. In particular, it seems a plausible way of explaining how the process of distinguishing an intended referent from a set of discourse entities might be achieved: the speaker need only construct a description which contains sufficient information to distinguish the intended referent from those other discourse entities which have the same or a greater activation level. In a complementary fashion, the hearer need only identify the most activated entity that satisfies the description provided by the speaker; if entities are considered as potential referents in decreasing order of activatedness, then the entire set of entities evoked in the discourse need not be considered.

There are two problems with this kind of approach. First, there are difficulties in implementing such a system. Other computational systems that have tried to take account of the effects on focus and topic of various syntactic constructions do so in exactly those terms: one construction is seen as shifting the topic, another is seen as putting an entity into high focus, and so on. Using discrete activation levels requires experimenting with numerical weightings, with the result that it is difficult to state significant generalities. Nonetheless, some researchers working in natural language understanding have tried approaches of this kind: see, for example, the work by Pollack and Waltz on the use of spreading activation in parsing [Pollack and Waltz 1985]. More recently, Alshawi [1987] has presented a detailed model where contextual factors are used to modify salience levels. However, no attempts have been made to use this approach in generation.

Second, an appealing use of the notion of concept activatedness would be in conjunction with the view that pronouns and non-pronominal definite referring expressions are on a cline. Under this view, we might place definite referring expressions of increasing complexity on a scale of information content, with the result that pronouns belong at one end of this scale, since they have minimal information content, specifying at most the number and gender of the entities they are used to refer to.

Unfortunately, the data does not bear out this position. If the claim were correct, then we would expect, for example, to be able to use *he* to refer to a male entity last mentioned arbitrarily long ago, provided no other male entities had been mentioned in the interim. However, examination of texts shows that pronouns are rarely used to refer to entities other than those mentioned in the preceding utterance. Although there are cases of LONG DISTANCE PRONOMINALIZATION, there are nowhere near as many as we would expect on the basis of this theory.[13]

2.3.5 Local and Global Focus

Above, we considered salience as a means to determining the context within which a definite determiner is used to refer to something uniquely. An alternative approach is to use the notion of a discourse model:

[13]In a sample of one hundred consecutive examples of pronouns from each of three very different texts considered by Hobbs, 98% of antecedents were found to be either in the same or previous sentence [Hobbs 1978:322–323].

> Uniqueness in a model rather than in reality is what controls the use and interpretation of definite descriptions. If a speaker is to communicate felicitously, then he must consider whether an entity will be unique in his listener's model.
>
> Johnson-Laird and Garnham [1980:377]

Grosz [1977], and subsequently Grosz and Sidner [1985, 1986], suggest a two-level approach to the modelling of discourse status, which intuitively corresponds to the distinction between short-term (or syntactic) and long-term (or semantic) memory.

The basic ideas here were expounded by Grosz [1977]. In this work, Grosz distinguished GLOBAL FOCUS and LOCAL FOCUS (also referred to as IMMEDIATE FOCUS). The general claim is that a distinction can be drawn between the hearer's memory for concepts, and the hearer's memory for linguistic form. Grosz claimed that local focus is more relevant to pronominal reference, since the syntactic form of the preceding utterance will still be available, whereas global focus is more relevant to definite noun phrase reference. This distinction can be likened to Chafe's distinction between GIVENNESS and DEFINITENESS. In another attempt to get at the intuitions involved, Sidner [1979] suggests that local focus is, essentially, 'what is being talked about'. Global focus, in Grosz's view, could further be decomposed into a collection of FOCUS SPACES, organised in a hierarchy that matched the hierarchical structure of the task-oriented dialogues she was dealing with.

Work in psycholinguistics lends support to the view that the hearer's memory for text consists of two parts, referred to as syntactic and semantic memory respectively. Guindon [1983] has explicitly used Grosz's distinction as a model in a number of experiments. Some work on anaphora in linguistics has also supported this general view: Hankamer and Sag [1976] draw a distinction between DEEP ANAPHORA and SURFACE ANAPHORA, corresponding to the distinction between semantic and syntactic memory. The most common anaphora are surface anaphora, which are always generated (and thus interpreted) with respect to the contents of syntactic memory. Other anaphora are deep anaphora, and are concerned with the 'semantic memory' part of the discourse model.

The Structures of Discourse Structures

Grosz's early work [Grosz 1977, 1981] was concerned solely with task-oriented dialogues. In more recent work in conjunction with Sidner [Grosz and Sidner 1986], the theory has been developed to apply to discourse in general. The basic ideas of the theory are as follows.

A discourse consists of hierarchically arranged DISCOURSE SEGMENTS. Each discourse segment may itself consist of a number of discourse segments, and so on. A series of utterances constitute a discourse segment if they together serve to realize a particular DISCOURSE PURPOSE. As yet, no algorithm has been developed for determining the structure of a discourse in terms of this segmentation. However, clue words and phrases such as *however*, *in any case*, *finally*, and so on, are generally agreed to be surface indicators of discourse structure (see, for example, Cohen [1984]).

Grosz and Sidner suggest that there are three different structures that we can identify within a discourse. First, there is the LINGUISTIC STRUCTURE of the discourse, as just described. Second, underlying this is the INTENTIONAL STRUCTURE. A discourse, as a whole, has a principal purpose or intention, which corresponds to the top level discourse as a whole. However, communicating this intention may require that a number of lower level discourse purposes be realized: corresponding to each of these purposes, there will be an embedded discourse segment. This decomposition may continue to an arbitrary depth.

The third structure of a discourse is ATTENTIONAL STATE. This defines the focus of attention in a discourse. Unlike the intentional structure, this does not mirror the linguistic structure of the discourse, but changes as a discourse proceeds, and is stack-like in nature. Corresponding to each discourse segment is a focus space, which contains the salient entities and relations introduced in that discourse segment. When a discourse segment is initiated, a new focus space is placed on the focus stack. When a discourse segment is completed, its focus space is removed from the stack. This mechanism attempts to capture the intuition that once some sub-discourse is complete, its contents are no longer relevant to the ongoing discourse.

Note that the theory is concerned only with the *structure* of a discourse: the implication

is that the principles are applicable across a wide range of discourse types, irrespective of the particular *semantic* relations holding between the discourse segments (e.g., whether a sub-discourse is an elaboration of some point or provides support for an argument). This is in opposition to the work of, for example, Reichman [1981, 1985], who suggests that the different semantic or rhetorical relations that can hold between CONTEXT SPACES affect the focus of the discourse in different ways.

The Interaction of Reference and Discourse Structure

Grosz and Sidner make the following claim (emphases added):

> Just as linguistic devices affect structure, so the discourse segmentation affects the interpretation of linguistic expressions in a discourse. *Referring expressions provide the primary example of this effect.* The segmentation of discourse constrains the use of referring expressions by delineating certain points at which there is a significant change in what entities (objects, properties, or relations) are being discussed. For example, *there are different constraints on the use of pronouns and reduced definite noun phrases within a segment than across segment boundaries.* While discourse segmentation is obviously not the only factor governing the use of referring expressions, it is an important one.
>
> [Grosz and Sidner 1986:178]

This claim, if true, obviously has considerable significance for the modelling of discourse. Grosz and Sidner do not state any explicit rules derived from their general claim; however, they draw attention to some reference phenomena that they interpret as lending it support. We will consider two examples from Grosz's earlier work on focusing in discourse [Grosz 1977], and two from the more recent work [Grosz and Sidner 1986].[14] In each case, I will assume that the structural analysis assigned to these discourses by the authors is correct. The structure assigned is indicated by means of the annotations on the left-hand side of the examples.

[14] As far as I am aware, these are the *only* examples the authors present as evidence of their claim.

DS0 1. A: I'm going camping next weekend. Do you have a two-person tent I could borrow?
2. B: Sure. I have a two-person backpacking tent.
DS1 3. A: The last trip I was on there was a huge storm.
4. It poured for two hours.
5. I had a tent, but I got soaked anyway.
6. B: What kind of a tent was it?
7. A: A tube tent.
8. B: Tube tents don't stand up well in a real storm.
9. A: True.
10. B: Where are you going on this trip?
11. A: Up in the Minarets.
12. B: Do you need any other equipment?
13. A: No.
14. B: Okay. I'll bring the tent in tomorrow.

Figure 2.3: The Tent Example (from Grosz [1977:2])

The Tent Example

In the discourse shown in figure 2.3, we have a conversation between two individuals. Grosz suggests that the discourse contains two discourse segments DS0 and DS1, where DS1 is embedded within DS0. Notice that the reference to *the tent* in utterance 14 refers back to the tent last mentioned in utterance 2, not the more recently mentioned tube tent which was introduced in utterance 5 and last mentioned explicitly in utterance 7.

Although an explanation could be given in terms of pragmatic factors—the speaker is unlikely to be offering to bring the other person's tent—Grosz offers an alternative explanation: because of the structure of the discourse, she suggests that the tube tent is no longer a potential referent by the time we reach utterance 14, since the focus space in which it was introduced (i.e., the focus space corresponding to DS1) has since been popped from the focus stack.

The Movies Example

In the text in figure 2.4, Grosz and Sidner suggest that the full noun phrase *a moving picture show* is used to refer back to *the "movies"* even though the antecedent was

1. The "movies" are so attractive to the great American public,
2. especially to young people,
3. that it is time to take careful thought about their effect on mind and morale.
4. Ought any parent to permit his children to attend a moving picture show often or without being quite certain of the show he permits them to see? ...

Figure 2.4: The Movies Example (from Grosz and Sidner [1986:183])

mentioned in the previous utterance; thus, a full noun phrase is used in a situation where we might have expected a pronoun to be used. Grosz and Sidner suggest that this is because the anaphor and its antecedent are in two different (sister) discourse segments.

There is something a little odd about the anaphoric relation being postulated in this example, since the supposed anaphor is an *indefinite* noun phrase. For this example to count as evidence for Grosz and Sidner's claim, it would be better if the supposed anaphor was unambiguously an anaphor in the normal sense. Fortunately for the theory, however, the use of full noun phrases in preference to pronouns for subsequent reference has been noted by others (notably Reichman [1981]). The text in (2-66), extracted from a letter of reference (slightly modified to protect the innocent), exemplifies the same phenomenon:

(2-66) I first met Dr. Smythe-Jones in 1979, while he was on leave from the University of the Yucatan. He spent some time as a visiting lecturer in Hamburg. ... During his time here, he proved to be a very innovative thinker.

Smythe-Jones's academic career may not have started very early, but he ...

If we take the paragraph break as a plausible indication of discourse structure, then the use of the proper name *Smythe-Jones* instead of a pronoun can be interpreted as emphasising that a new discourse segment has begun. As a corollary, use of a pronoun would then be interpreted as an indication that the current discourse segment remains open.

18. A: The two screws are loose, but I'm having trouble getting the wheel off.
19. E: Use the wheelpuller. Do you know how to use it?
20. A: No.
21. E: Do you know what it looks like?
22. A: Yes.
23. E: Show it to me please.
24. A: OK.
25. E: Good. Loosen the screw in the center and place the jaws around the hub of the wheel, then tighten the screw onto the center of the shaft. ...

Figure 2.5: The Screw Example (from Grosz and Sidner [1986:186])

The Screw Example

In the discourse shown in figure 2.5, we have a dialogue between an apprentice (A) and an expert (E). Here, neither of the screws mentioned in utterance 18 is interpreted as the intended referent of *the screw in the center* in utterance 25; instead, the intended referent is part of the wheelpuller introduced in utterance 19 (thus the referent is never actually introduced into the discourse explicitly, but only implicitly). The claim here seems to be that since the intended referent is (by association with the explicitly mentioned wheelpuller) in the current focus space, potential distractors in focus spaces further down the stack are not considered. Note, however, that we do not need to appeal to discourse structure here: straightforward recency of mention of an associated entity would provide an equally plausible explanation.

The Compressor Example

Finally, we have another example from Grosz's earlier work, shown in figure 2.6. Here the pronoun *it* in the final sentence of the discourse refers to the compressor, which was last mentioned back at the beginning of the discourse. The implication here is that the entities introduced in the intervening discourse are no longer considered as 'potential distractors' because the focus spaces in which they were introduced are no longer on the focus stack. This is then essentially the same claim as that made for the tent example discussed above.

E: Good morning. I would like for you to re-assemble the compressor.

...

E: I suggest you begin by attaching the pump to the platform.

[...other subtasks]

E: Good. All that remains then is to attach the belt housing cover to the belt housing frame.

A: All right. I assume the hole in the housing cover opens to the pump pulley rather than to the motor pulley.

E: Yes, that is correct. The pump pulley also acts as a fan to cool the pump.

A: All right, the belt housing cover is on and tightened down.

[30 minutes and 60 utterances after beginning]

E: Fine. Now let's see if it works.

Figure 2.6: The Compressor Example (from Grosz [1977:23])

Note, however, that this example can also be explained as deixis, since the compressor is present when the pronominal reference is made.

Summary So Far

As should be obvious from the above examples, the general claim is that the *structure* of a discourse—irrespective of its content—determines, in our terms, the context with respect to which an intended referent must be distinguished.[15]

However, as we noted, in three of the four examples there are alternative explanations for the referential behaviour observed: pragmatics in the case of the tent example, recency in the case of the screw example, and deixis in the case of the compressor example. Also, as noted above, there is something odd about the anaphor/antecedent relation postulated in the movies example. The tent example is perhaps the most convincing; however, whereas the others are drawn from real discourses, this particular example is hand-constructed (Grosz [1978:230]).

Thus, the evidence offered by Grosz and Sidner for the claim they make is not as strong as we might wish. There are other problems: for example, little is said about the

[15]It will not have escaped the notice of many readers that the model proposed by Grosz and Sidner is reminiscent of the variable scoping mechanisms used in structured programming languages.

constraints upon subsequent reference to entities which have only appeared in focus spaces that are now closed. Grosz and Sidner only claim that the entities in the popped focus space are no longer referrable to 'in the same way', but it is not clear what this really means.

Nonetheless, the notion that the universe of referents in a discourse can be partitioned in some way is a popular one. The same underlying ideas surface in several recent approaches to discourse structure within computational linguistics: Reichman's context space theory [Reichman 1978, 1981, 1985] makes similar claims for the effect of discourse structure on reference, as does the linguistic discourse model (LDM) due to Polanyi and Scha [Polanyi and Scha 1984; Polanyi 1985, 1986].[16] Within linguistics, Grimes [1982] has made use of a notion of reference spaces which is essentially the same as Grosz and Sidner's approach; and Linde [1979] demonstrated how the use of referring expressions within discourses describing an apartment suggests that discourse structure has an important role to play. Other work has focussed on what we might call logical structure rather than rhetorical structure in discourse; thus, Karttunen's suggestions for dealing with hypothetical entities [Karttunen 1976], Kamp's discourse representation theory (DRT) [Kamp 1981], and Fauconnier's work on mental spaces [Fauconnier 1985] are broadly compatible in that they view structural concerns as important in restricting the context of interpretation of referring expressions. From a more psychological perspective, Sanford and Garrod [1981] suggest that memory consists of four independently addressable components, called explicit focus, implicit focus, long term semantic memory, and long term text memory. Reichgelt [1986] suggests a model of the human language processor that involves embedding of models within models. Chafe [1977, 1979] suggests a hierarchical model of memory whose units are the memory (corresponding to a story), the episode (corresponding to a paragraph), the thought (corresponding to a sentence) and the focus (corresponding to a phrase or clause).

One possibility we have not considered above is that of *combining* the two approaches to relative salience that we have just discussed: that is, we might adopt a structured discourse model as a gross characterisation of relative salience, but still make use of a discrete notion of activatedness within individual parts of this model. To some extent

[16]The theories developed by these researchers vary in their emphasis and complexity: Grosz and Sidner's is the simplest of the three, since it attempts to abstract away from any particular semantic or rhetorical relations in discourse. Apart from a claim about pronominalization made by Reichman, the three theories do not make substantially different claims about the effects of discourse structure on reference.

this idea is implicit in Grosz and Sidner's work, for example, since (as we saw in section 2.2) a notion of sentential topic or *center* is still required in their theory in order to account for pronominalization. We might suggest, then, that in any given discourse segment, those entities referred to can be ordered in terms of their relative salience *within that segment.* Pursuing this possibility is beyond the scope of the current work.

2.4 One-Anaphora

Not all anaphoric forms pick out individuals already introduced into the discourse. The *one*-anaphoric form is typically used to identify an individual which is of the same *type* as some other individual mentioned in the discourse, and as such is sometimes referred to as DESCRIPTIONAL anaphora.

2.4.1 Different Uses of *One*

A number of different uses of *one* are distinguished in the literature. Of these, the principal three are known as SUBSTITUTE *one*, the indefinite personal pronoun, and the cardinal number *one*. There are also a number of more minor uses of the form. We provide examples of each type below in order to clarify the distinctions.

The word *one* can take the place of both whole and part noun phrases, as in (2-67) and (2-68) respectively:

(2-67) a Judy bought *a large green wombat basket.*
b Mary bought <u>one</u> too.

(2-68) a Judy bought *a small blue wombat basket.*
b Mary bought a large green <u>one</u>.

In formal speech, *one* is also used as an indefinite personal pronoun:

(2-69) <u>One</u> should never put <u>one</u>'s cat out at night

This use is sometimes also referred to as GENERIC *one*.

The third major use of *one* is as a cardinal number, as in

(2-70) I have <u>one</u> cat, although I'd like to have more.

Apart from these three principal uses of *one*, there are other, less frequent, uses. *One* can be used as an alternative form of the indefinite article, as in

(2-71) <u>One</u> day I'll come and visit you.

(2-72) <u>One</u> politician is as bad as another.

We will now go on to examine substitute *one* in more detail. Our primary interest is in substitute *one*, and so we will go on to consider this use in more detail.

2.4.2 What Does 'One' Substitute For?

First, we should note various aspects of the use of the *one*-anaphoric forms. In each case, the head noun which is replaced must be a count noun:

(2-73) a *These biscuits* are stale.
b Get some fresh <u>ones</u>.

(2-74) a *This bread* is stale.
b Get some fresh.

Notice that it is *not* necessary that the syntactic structure of a *one*-anaphoric noun phrase match that of its antecedent:

(2-75) a Do you have *any bullets made of platinum*?
b No, but I have <u>some leaden ones</u>.

However, there do appear to be *semantic* constraints on possible combinations:

(2-76) a Do you have *a milk bottle*?
b ?No, but I have <u>a red one</u>.

One-anaphora appears transparent to number: a plural *one*-anaphor can be used with a singular antecedent, and *vice versa*.

(2-77) a Jon bought *a grey T-shirt*.
b Marc preferred <u>the brighter ones</u>.

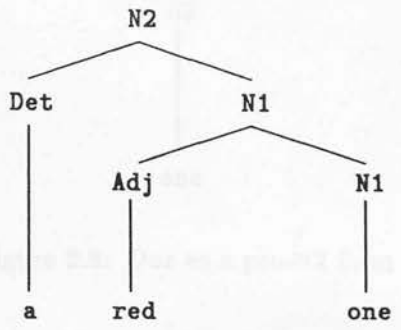


Figure 2.7: *One* as a pro-N1 form

(2-78) a Jon bought *some coloured T-shirts.*
b Marc preferred the green one.

Under what we might call the STRUCTURAL SUBSTITUTION APPROACH to *one*-anaphora, *one* is seen as a substitute for a syntactically-defined element. As we saw above, the word *one* appears to substitute for both whole noun phrases and for part noun phrases, where by a part noun phrase we mean a head noun with zero or more modifiers. That there are in fact two distinct types of substitutions going on here becomes more obvious when we consider the plural versions of the two cases:

(2-79) a Does anyone have *any green wombat covers*?
b Yes, I have some.

(2-80) a Do you want *the green wombat covers*?
b No, I want the blue ones.

Most syntactic analyses would have *one/ones* function as, in X-bar terms [Jackendoff 1977], N1 pro-forms. Thus, they must always appear with a determiner. The noun phrase in example (2-81) then has the structure shown in figure 2.7.

(2-81) a Has anyone got *a blue jumper*?
b No, but I've got a red one.

There are differing analyses of the second pro-form. For Leech and Svartvik [1975] and Quirk *et al* [1985], for example, both pairs of pro-forms are substitutions: as above, the

Figure 2.8: *One* as a pro-N2 form

one/ones pair are pro-N1 forms, whereas the *one/some* pair are pro-N2 forms. Thus, the *one*-anaphoric noun phrase in example (2-82) has the structure shown in figure 2.8.

(2-82) a Who has *a copy of Cosmopolitan*?
b I have one.

However, Halliday and Hasan [1976] reserve the term 'substitute' for the *one/ones* pair, and prefer to see the *one/some* pair as instances of the indefinite article [Halliday and Hasan 1976:101]. Thus, in (2-82), *one* functions as a determiner where the rest of the noun phrase has been elided, resulting in the structure shown in figure 2.9.[17] This seems more plausible, since we also parallel usage of cardinal numbers, as in the following:

(2-83) a Glyn brought Mike a debugging aid.
b Jo brought Henk two.

Extensive discussions of *one*-anaphora can be found in the linguistics literature. In particular, see Halliday and Hasan [1976:91-105], Leech and Svartvik [1975, sections 394-396], Quirk *et al* [1985, sections 6.54-56 and 12.15-16], Gazdar *et al* [1985:130], Baker [1978:324-347], and Hornstein and Lightfoot [1981:16-23]. The major computational treatment is that of Webber [1979]: we will use some of the basic ideas from that work in our own treatment of *one*-anaphora presented in chapter 5.

[17]Strictly speaking, we should perhaps retain the use of the term *one*-anaphora only for the substitution pair *one/ones*; however, we shall, for convenience, continue to use the term also to refer to the use of the *one/some* pair.

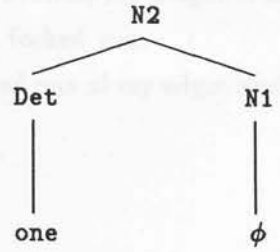


Figure 2.9: *One* as a determiner

2.5 Previous Approaches to the Generation of Referring Expressions

This section looks at the ways in which existing language generation systems approach the problem of referring expression generation. We describe, in detail, the relevant aspects of three of the most well-known natural language generation systems: Anthony Davey's PROTEUS [Davey 1978], David McDonald's MUMBLE [McDonald 1980a], and Douglas Appelt's KAMP [Appelt 1982, 1986]. We then examine briefly the approaches taken by a number of other systems.

2.5.1 Davey's PROTEUS

Davey's PROTEUS [Davey 1978] was one of the first significant language generation systems to appear. The program describes games of noughts and crosses (tic-tac-toe), and addresses aspects of both deciding what to say and how to say it (cf. the discussion in chapter 1 of this thesis). Within this limited domain, the program is capable of generating quite sophisticated output. Here is a description of a game produced by the program (from Davey [1978:17]):

(2-84) The game began with my taking a corner and you took an adjacent one.
I threatened you by taking the corner adjacent to the one that you had just taken, but you blocked my diagonal and threatened me.
I blocked yours and forked you.
Although you blocked one of my edges and threatened me, I won by completing the other.

An Overview of PROTEUS

The input representation to the program is a list of moves representing the history of a game: each element in the list is simply a pair, indicating a player and the number of a square taken by that player. So, for example, we have the following as a representation of the first three moves in a game:

(2-85) `[[<Proteus> 7] [<ACD> 5] [<Proteus> 3]]`

On the basis of the significance of each move, PROTEUS then decides how many moves can be described in a given sentence. Since, in the above example,[18] the first move by each player is tactically insignificant, PROTEUS decides that all three of these moves are to be realized in a single sentence. This results in the following intermediate representation, which incorporates both some assessment of the significance of the moves (in this case, the fact that the first move is the start of the game) and some information required for building the sentence (the required punctuation and conjunctions):

(2-86)
```
[[<Proteus> <square 7> start <game> take <square 7>],
[<ACD> <square 5> take <square 5>], and
[<Proteus> <square 3> take <square 3>].]
```

The first part of this sentence is then realized as

(2-87) I started the game by taking a corner, ...

How PROTEUS Generates Referring Expressions

Because of the simplicity of the noughts and crosses domain, the kinds of objects that are talked about in Davey's program are very limited, belonging to three 'broad' types: the game itself, the players, and the board and its parts. This last type is subcategorised

[18] This example is described in considerably more detail in Davey [1978:154–164].

as a number of 'narrow' types: corners, edges, lines, squares, and diagonals [Davey 1978:134]. For each type of object, the program has available only one lexical item [Ritchie 1985], thus removing any need for a genuine process of lexical choice.

In order to generate references to objects, the program keeps track of two lists of objects previously referred to (cf. Davey [1978:134–139]):

- PREVREF (the PREVious REFerents list) contains an entry for each item mentioned in the current sentence and the preceding sentence. Each entry specifies the referent, its narrow type, and the syntactic features of the expression used.
- CANLIST (the Currently Active Node List) contains an entry for the referent of each item referred to in the current NP being constructed.

When a reference to an entity has to be constructed, the program tries first to use a proper name, then a pronoun; if neither of these strategies is successful, it builds a definite noun phrase.[19]

Pronominalization

PROTEUS considers a reference to be pronominalizable if the intended referent is the system or the user (in which case a personal pronoun can be used), or if the intended referent was mentioned recently and there are no distractors. An entity is said to have been mentioned recently if it is present in PREVREF or CANLIST; a distractor is any entity which has been mentioned since the intended referent was mentioned, and which has the same number, person and animacy as the intended referent.

Non-pronominal Reference

If a pronoun is ruled out, the algorithm used to generate a non-pronominal reference is then as follows, where x is the intended referent.

First, a description D is constructed for x. To do this, the program finds the narrow

[19] The description of Davey's algorithm presented here is based on a rational reconstruction of Davey's own, as presented by Ritchie [1985].

type of the smallest set including x; this is then restricted to denote a subset of the EQUIVALENCE CLASS[20] of x but still including x: i.e., D is not true of any items outside the equivalence class of x. If this is not possible due to x being a set of items with no shared equivalence class, x is then split according to these separate classes, each is described individually, and the descriptions are conjoined.

The description construction process involves a number of heuristics:

- if there is a set S such that the equivalence class of x is a subset of S, and the complement of the equivalence class of x with respect to S is P, and, for all objects in P, that object has been mentioned recently (implicitly or explicitly), then the adjective *other* can be added to the description;
- if a square is empty (i.e., not taken), then this may be used to distinguish it in the computation of a description, without adding anything explicit to the description;
- if an object has been *implicitly* mentioned recently, then the adjective *same* can be added to the description: a line, for example, is considered to have been implicitly mentioned recently if two squares in it have been mentioned recently.

Once a description has been constructed, the following steps are then applied:

- If there is some other entity y present in PREVREF which has the same narrow type as x, and if there is no other entity z in PREVREF and mentioned since y which has the same broad type as x, then the head of the description D can be reduced to *one*.
- If the description contains the adjective *other*, the head can be deleted completely.
- If D is true of exactly the items in x, then a definite determiner can be used; otherwise an indefinite determiner is used.

The first step here determines whether the context permits reduction of the head noun, licensing references like

(2-88) You took a corner and I took the *other*.

[20]In any given state of the game, each square (or line) p on the board will have an equivalence class, which is defined to be the set S of squares (or lines) to which p would map under any transformation of the board which preserved the symmetry of the configuration, including ownership of squares.

while prohibiting similar references in contexts like

(2-89) *You took a corner and threatened my line and I took the *other*

Once a referring expression has been constructed, x is added to PREVREFS.

2.5.2 McDonald's MUMBLE

David McDonald's MUMBLE [McDonald 1980a][21] is perhaps the most well-known language generation system, and is recognised as having the widest grammatical coverage of existing systems.

An Overview of MUMBLE

Unlike PROTEUS, MUMBLE is purely a tactical component: it is concerned with getting from the content of what McDonald variously calls a *message* or *realization specification* to a linguistic realization of that message. MUMBLE assumes that all decisions as to the *content* of what is to be said have already been decided by some other part of the system (the 'expert program').

Each realization specification typically contains more than just propositional content: it may also specify, for example:

- which speech act is being performed;
- what can be assumed, and what must be stated; and
- any impositions of ordering, highlighting and word choice.

Thus, a realization specification might specify that a particular element of the message is to be made the focus of the utterance. Messages need not be translated as single sentences: in the case of messages which are too large to be translated as a single clause, sequencing information may also be specified.

[21] McDonald's thesis [McDonald 1980a], which contains the most complete description of MUMBLE, can be difficult to obtain. Overviews of the system can be found in [McDonald 1981a] and [McDonald 1983]; the subsequent reference strategies are described in [McDonald 1978]; and some discussion of requirements for a dictionary can be found in [McDonald 1981b]. The background to the development of MUMBLE can be traced further in [McDonald 1977, 1979].

MUMBLE is designed to work with a number of different input representations, the differences being ironed out by tailor-made 'dictionary functions'. The dictionary has to be rewritten for each new expert program that MUMBLE is to work with. It contains an entry for each 'concept, name, structure, process or other entity' [McDonald 1977:18] that the program might want to talk about: for each such entity (or *message-element*), it provides the correspondence between that entity's internal representation and its possible English phrasing.

A distinction needs to be drawn between the representation used for the expert program's knowledge base, and that used for the message to be conveyed. In some cases, the simplest description of the information to be communicated *is* the data structure used to represent that information in the system's knowledge base; however, in general, it is necessary to embed the knowledge base's data structures within relations that are particular to the needs of natural language communication. Thus, given the following data structure as input

```
(message
    (sequence
       (macbeth (murder (duncan)));    "murder-ma"
       (macbeth (become (king)))  ;    "ma-become-king"
       (lady-macbeth (persuade    ;    "persuade-ma"
                     (macbeth
                       (action murder-ma))))
       (lady-macbeth (ambitious)));    "ambitious-lm"
    (time-frame
       (before-time-of-speech))
    (focus
       (macbeth))
    (ancillary-facts
       ((murder-ma (motive (ma-become-king))))
       ((persuade-ma (purpose (cause (murder-ma)))))
       ((lady-macbeth (modifies (ambitious-lm))))))
```

MUMBLE will produce the output

(2-90) Macbeth murdered Duncan in order to become king. He was persuaded to do it by Lady Macbeth, who was ambitious.

How MUMBLE Generates Referring Expressions

MUMBLE takes many of the ideas embodied in PROTEUS a step further. McDonald distinguishes two kinds of reference that can be made within a discourse: *initial reference* and *subsequent reference*. We use initial reference when we refer to an object for the first time in a discourse, and we use some form of subsequent reference on subsequent occasions.

The lexical entries MUMBLE uses to 'realize' internal objects as linguistic items are much more complex than those used by PROTEUS: whereas PROTEUS, as we saw, had a simple one-to-one mapping from types of objects to the lexical items used to realize them, MUMBLE makes use of procedural entities called COMPOSERS which take into account various aspects of the context in order to produce an appropriate linguistic string.

If an object is being mentioned for the first time, it is realized by a process referred to as *mainstream realization*: this means that the information held in the lexical entry is used as is. However, once an object has been mentioned, it is added to the REFERENCE LIST. Subsequent occurrences of that object in the message stream then cause the program to invoke one of various SUBSEQUENT REFERENCE STRATEGIES: either a pronoun can be used, or some alternative reference strategy can be adopted. These strategies may involve modifying the mainstream realization process (typically, repeating some cut-down form of the original reference). The mainstream realization process involves several component steps, whose behaviour can be modified on input of suitable information to the procedure. One such modification results in the modifiers from a noun phrase being dropped. As a catch-all, if no subsequent reference strategy can be used, then mainstream realization is used by default.

Deciding whether to use a pronoun

The PRONOMINALIZATION DECISION is composed of two stages: the first of these involves simple checks, and is therefore computationally easier than the second. Thus, if the first stage decides that a pronoun cannot be used, the heavier cost of the second stage can be avoided.

The first stage checks to see if an element has already been marked for pronominalization (this relates to the rhetorical uses of subsequent reference: the 'expert speaker' may have already decided that pronominal reference is required or ruled out). It also checks the ONTOLOGICAL TYPE of the message element, i.e. whether it is *referential* or *descriptive*. If it is the latter, pronominalization is ruled out since this may cause information loss. Finally, there is a 'syntax constraint' which ensures that only message elements that might be realized as noun phrases are considered for pronominalization.

If the first stage of the decision process does not rule out use of pronominalization, then the second stage of the process comes into operation. In summary, the process involves two steps: first, the relationship of the present instance of the message element to previous instances is analysed, to see whether a pronoun would be permissible; then, any nearby references to entities which are POTENTIAL DISTRACTORS, in the sense that they could cause a pronoun to be ambiguous, are taken into consideration.

A number of heuristics determine whether or not pronominalization is advisable, by taking into account the relation between the current instance of the message element and its previous realizations. These heuristics cast votes of three strengths: necessary, strong evidence and weak evidence. Some of the factors taken into account are

- whether the previous mention of the intended referent was introduced in a subordinate context: if it was, this votes strongly against the use of a pronoun;
- whether the previous reference was pronominalized: this votes strongly for a pronoun;
- if the intended referent was not the most recently mentioned element, this votes weakly against pronominalization;
- if the intended referent is the potential actor focus (after Sidner [1979]), this produces a strong vote for a pronoun;
- if the intended referent was referred to by the most recent NP, this votes weakly for a pronoun; and
- whether the previous reference *precedes and commands* the new reference:[22] if so, this demands a pronoun.

[22]This is a syntactic constraint on pronominalization, of the kind mentioned earlier in this chapter. McDonald attributes the constraint he uses to Langacker.

In addition, McDonald suggests that it is stylistically more acceptable to use pronouns in parallel constructions, and that the discourse status of the entity in question is of importance: he states that subsequent references to the current focus are always pronominalized, although it is not clear exactly what use McDonald makes of focus.

The above is not enough to tell whether pronominalization is safe: there is also the possibility of distracting references. Here semantic and pragmatic properties of the objects involved have an effect. MUMBLE builds a set of POTENTIAL DISTRACTORS: these are the 'important' elements in the current sentence and the last sentence, as well as the current focus. Some elements of this set may then be dismissed on syntactic grounds. Then, a function called `distinguishable-kinds` is responsible for eliminating any entities the hearer is expected to be able to distinguish.[23] If there is any possibility of ambiguity, MUMBLE tries to second-guess the hearer's interpretation.

Subsequent Reference

In MUMBLE, when a reference to an object is made, a certain amount of information about that reference is stored. This information is stored in two places: we have both a DISCOURSE LIST and a set of DISCOURSE RECORDS. The discourse list is merely a list of all the message elements that have been mentioned in the discourse so far; a discourse record is

> ...a vector of just those properties which, from the point of view of later routines such as the pronominalization heuristics, are sufficient to characterize that instance of the message element in the discourse.
>
> [1978:68]

McDonald [1978] implies that this latter information is actually incorporated into the more permanent information in the dictionary entries themselves:

> ...when the generation component realizes an instance of an object as a phrase, it can add an annotation to [the entry] marking what kind of phrase was selected, where in the text this occurred, what the immediately dominating clause was at the time, and so on. The next time there is an instance

[23] The operation of this function is unclear from McDonald's description [McDonald 1980a:220–221]: the function is said to appeal directly to the speaker program.

> of that same object the annotation can be found and used to help decide what kind of subsequent reference should be made.
>
> [1978:65]

Certain aspects of syntactic context are calculated and stored as a message is translated: for example, current-main-clause, current-verb-phrase, and so on. These variables can then be used in computations such as the decision to pronominalize.

Non-Pronominal Subsequent Reference

MUMBLE views the process of non-pronominal subsequent reference as consisting of a set of conventions that govern the ways in which the amount of information conveyed by a referring expression can be minimized while still making identification of the referent possible. The particular form of non-pronominal subsequent reference chosen depends on two things. First of all, there is the reason that the message element could not be pronominalized: this is read from the list of pronominalization heuristics, although McDonald does not explicitly specify how this information is used. The second thing that the chosen form depends upon is the initial realization of the message element: this is obtained from the discourse history. Again, it is not clear exactly how this information is used.

The general strategy adopted by MUMBLE is to omit all the modifiers initially used, leaving the head noun (which describes the object's class) and a definite determiner.

2.5.3 Appelt's KAMP

Douglas Appelt's KAMP [Appelt 1982, 1986] has its roots in Cohen's work [Cohen 1978] on the formalisation of speech act theory. In line with this, Appelt's principal conviction is that language is best viewed as plan-based behaviour which involves reasoning about the effects of utterances on the hearer's state of mind, with the result that planning is the best approach to language generation. To this end, KAMP produces plans consisting of both physical and linguistic actions.

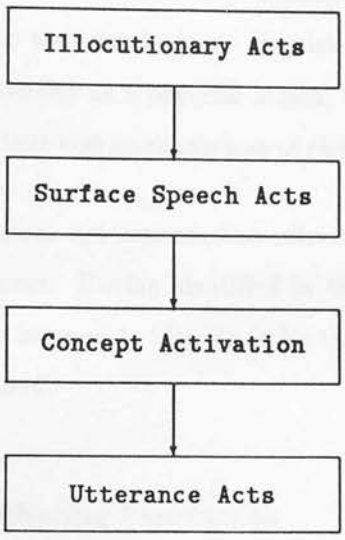


Figure 2.10: A Hierarchy of Language-Related Actions (from Appelt [1982:10])

An Overview of KAMP

Appelt's work is unlike much other work in language generation, in that it disputes the view that the decisions of what to say and how to say it can be separated (recall the discussion in chapter 1 of the present work). Appelt views the process of language generation at various levels of abstraction, in a hierarchy as shown in figure 2.10.

KAMP allows interaction between the lowest levels of the production process and the high level decisions. The highest level actions decided upon are illocutionary acts, such as informing or requesting. These do not specify anything about the ultimate linguistic realization of the utterance being planned.

At the next level, appropriate surface speech acts are selected. These are abstract representations of sentences with particular syntactic structures: this is the level at which linguistic knowledge comes into operation. At this stage, KAMP decides whether the basic structure of the utterance being planned is declarative, interrogative or imperative.

Next, KAMP plans CONCEPT ACTIVATION ACTIONS, involving descriptions that are mutually acceptable to the speaker and the hearer. This is where KAMP decides upon the particular propositional acts to be carried out. Concept activations are more abstract than acts of reference, since they need not be linguistic; and so a concept activation might also be partially expanded as a physical action, such as pointing. This kind of thing might happen where there was no simple way of picking out an object linguistically.

At the final stage, these actions are expanded as *utterance acts*, which involve specific words and syntactic structures. Having identified in the previous stage exactly what aspects of an object need to be used to identify it for the hearer, this stage decides the exact form of words to be used.

How KAMP Generates Referring Expressions

Where the realization of the act is linguistic, concept activation actions usually lead to the planning of some description of the object to be referred to. Appelt does not specify the algorithm he uses in detail, but it appears to be something like the following, where x is the intended referent:

- if x is mutually believed to be in focus, and x is pronominalizable, then use a pronoun;
- otherwise, build a minimal description D for x.

Appelt does not go into detail as to how the program decides whether or not a given referent is mutually believed to be in focus, and how it decides whether or not a reference is pronominalizable, but he states that the rules involved are adapted from Sidner's [1979] rules for keeping track of focus and resolving anaphora in discourse comprehension.

In KAMP, the notion of focus is central to the generation of referring expressions, and is intricately tied into the notion of concept activation. The result of a successful act of pointing or description is that the concept in question is held to be active for both the speaker and the hearer, i.e.:

(2-91) Active(S,H,C)

This concept then becomes a potential focus. Appelt does not specify how an activated concept becomes the immediate focus.

Appelt does provide a more detailed explanation of how KAMP builds minimal descriptions. A description D of an object x consists of a conjunction of one or more descriptors that apply to that object. A description is said to be adequate for a speaker S to activate a concept C for hearer H if

(2-92) MutuallyBelieve(S,H,$\forall x D(x) \supset x = C$)

i.e., if the speaker and hearer both believe that the given description applies only to the concept in question.

Descriptor selection begins by choosing a basic-level descriptor. This is a notion originally defined by Rosch *et al* [1976]:

> Basic level descriptors ... are descriptors that describe an object as belonging to a category that is assumed by the speaker and hearer to be the 'level of abstraction at which the organism can obtain the most information with the least cognitive effort'. For example, ***chair*** is the basic-level descriptor of objects in an abstraction hierarchy that includes ***furniture*** as a superordinate and ***recliner*** as a subordinate.
>
> [Appelt 1982:156]

Provided this 'default predicate' can be shown to be mutually believed by the speaker and the hearer, it is automatically incorporated into the description being built. KAMP then has to prove that the description being constructed is sufficient to identify the intended referent. Like PROTEUS, KAMP tries to generate a minimal description; however, whereas PROTEUS tried to disambiguate a reference with respect to all objects in the domain, KAMP's primary goal is to distinguish the object from all the other objects currently in focus. KAMP may also augment minimal descriptions where necessary: additional informing actions, for example, may be incorporated into the description being generated. This notion of ACTION SUBSUMPTION is not addressed by the other programs we have considered; it allows KAMP to generate descriptions which serve multiple purposes. Thus, the sentence

(2-93) Tighten the screw with the long Philips screwdriver

can realize several illocutionary acts, like a REQUEST to tighten the screw and an INFORM that the tool for tightening the screw is the long Philips screwdriver. In a situation where the speaker knows that the hearer doesn't know that a particular screwdriver is a Philips screwdriver, the utterance could in that case also serve to inform the hearer that the long screwdriver is a Philips screwdriver. So, not only is it that the same surface speech act may realize different types of illocutionary acts depending on the context, but it can realize a different number of illocutionary acts in different contexts [Appelt 1982:115].

Note that if descriptors have been added to the concept activation being planned as a result of action subsumption, then pronominalization cannot take place.

> In the case of indefinite and attributive reference, the description is already specified as part of the concept to be activated. For example, activating the concept $\iota x : P(x)$ constrains the speaker to use the description P.
>
> [Appelt 1982:122]

This is essentially the same as McDonald's check for ontological type, as outlined in the previous section. It is not clear whether this approach would rule out the possibility of a subsequent decision to use *one*-anaphora in such cases.

2.5.4 Other Language Generation Systems

Of course, there have been many other natural language generation systems which have had something to say about the generation of referring expressions. In most early systems, noun phrases were generated by instantiating simple patterns (see, for example, [Slocum 1975, 1978]): this was usually adequate given the other concerns of the systems in question. However, a number of systems apply a little more sophistication to the task: this section summarises the relevant details of a number of these.

Winograd's SHRDLU

Although Winograd's program [Winograd 1972] was not first and foremost a natural language generation system, it was capable of generating responses to questions about the blocks world domain it dealt with. Many of the responses provided by the program

are created using simple patterns triggered by various aspects of the question [Winograd 1972:163–166], but more sophistication is required when objects are to be described.

Each object is described in terms of its basic classification (e.g., #BLOCK, #BALL or #PYRAMID) and its colour and size. If the resulting description uniquely describes the intended referent, the determiner *the* is added; if the description is not unique, then what happens next depends upon whether a specific or nonspecific description is required. In the latter case, the determiner *a* (or *an*) is used; in the former case, more information is added in the form of a relative clause beginning with either *which supports* or *which is to the right of*. The program does no more at this point, even if the description is still not unique [Winograd 1972:166–168].

In addition to the above strategies, the program can also use limited *one*-anaphora and embodies a simple pronominalization heuristic [Winograd 1972:168–169].

McKeown's TEXT

McKeown's TEXT [McKeown 1982, 1986] is essentially a strategic component, and is therefore not itself concerned with the actual construction of surface noun phrases. However, her system is of interest because it explicitly makes use of Sidner's notion of focus [Sidner 1979] in generation. In fact, TEXT uses focus in the generation of pronouns in two ways. Once the content of an utterance has been decided upon, the program implements something like the inverse of Sidner's pronoun interpretation algorithm in order to determine whether pronominalization is possible. However, it is also on the basis of focus that the program decides *what* to say next. It can be argued that it is this second use of focus that is the most appropriate one in a language generation system: thus, the function of focus in a generation system can be seen as quite different from its use in an interpretation system. For the speaker, focus may play a more important role in deciding *what* to say, than in deciding *how* to say it.

HAM-ANS

HAM-ANS [Jameson and Wahlster 1982] emphasises the importance of user-modelling in the context of dialogue, and particularly in the generation of elliptical utterances. In

order to produce responses which are unambiguous but at the same time not too verbose, the system makes use of an ANTICIPATION FEEDBACK LOOP to simulate the user's understanding of a response. The first step in this process involves mapping the semantic representation of the utterance to be generated into a structure which has the property that each subpart of the structure corresponds to an element that might be elided (i.e., there is a very close correspondence between the semantic and syntactic structures). The system then determines the smallest substructure that realizes information not already contained in the question to which the utterance is a response. Once this has been identified, the anticipation feedback mechanism is used to check whether the hearer will be able to interpret this minimal response appropriately; if not, the next most informative response is considered. HAM-ANS is unique amongst the systems described here in making use of a sophisticated representational formalism that permits generation of both specific and non-specific indefinite noun phrases. The anticipation feedback mechanism is also used by the system to check the use of pronominal forms [Busemann 1984].

Granville's PAUL

In work which takes as its starting point the analysis of cohesion provided by Halliday and Hasan [1976], Granville's PAUL [Granville 1984] system chooses between pronominalization, superordinate substitution and definite noun phrase re-iteration when making subsequent reference to entities in discourse. Each form of reference has a STRENGTH OF ANTECEDENCE RECOVERY: thus, definite noun phrase re-iteration has the greatest strength of antecedence recovery, and pronominalization has the least. Given the location of the antecedent noun phrase in the text and some syntactic information, PAUL then determines the STRENGTH OF POTENTIAL ANTECEDENCE of the element to be referred to (using heuristics which are similar in concept to those used by McDonald's MUMBLE [McDonald 1980:218–220], as described earlier in this section). Mappings between these two measures then determine the best means of subsequent reference to use.

Houghton

Houghton [1986] describes a system in which two actors co-operate to achieve some goal. The micro-world used in the system consists of two agents, Fred and Doris, who are at either side of a door, and must communicate with each other in order to change the positions of objects in the world, including themselves. The approach Houghton takes is based on earlier work by Power [1974], and bears certain similarities. It also embodies a sizeable grammar fragment that is based on GPSG.

For the generation of referring expressions, Houghton explicitly adopts Prince's [1981] taxonomy of given/new information (discussed earlier in this chapter). This is implemented by means of a systemic network (as is the mechanism by means of which the content of a referring expression is determined), traversal of which in conjunction with information regarding the discourse status of the intended referent results in the appropriate decisions about the form of the referring expression.

XTRA

XTRA is a system whose purpose is to act as an intelligent interface to an expert system. The program is intended to generate both referring expressions and pointing gestures, the latter being made by means of positioning the cursor on the display [Reithinger 1986]. The system makes use of a simple pronominalization rule (only references to entities last mentioned in the previous sentence may be pronominalized), although the stated aim [Reithinger 1986:6] is to make use of a context-space model akin to that presented by Reichman [1985]. In the context of providing the user with advice on filling in forms displayed on the screen, it is also intended that the system will be able to use descriptions of regions of the screen in conjunction with pointing gestures to identify the regions of interest.

Novak's NAOS

Novak [1987] describes the problems of generating referring expressions in the context of a dynamically-changing street scene. When constructing a referring expression, NAOS

first considers using either a pronoun or a definite noun phrase that contains only static properties of the intended referent. However, if neither of these strategies results in an unambiguous reference, the system then uses a history of the events in which the various objects in the domain have participated to construct an appropriate reference containing a restrictive relative clause. This results in noun phrases like that in (2-94):

(2-94) the car which overtook the truck

If the relation expressed in the relative clause makes reference to another entity, only the static properties of that entity are used in referring to it (thus preventing recursively embedded relative clauses).

2.6 Summary

In this chapter, we have looked at the wide range of devices natural language offers for carrying out anaphoric reference, and we have surveyed existing approaches to the generation of referring expressions. In this section, we specify more precisely the particular phenomena that we will address in the remainder of the thesis.

In what follows, we will not be concerned with non-referential uses of noun phrases. In common with other existing generation systems, we will be concerned only with definite and specific indefinite reference; we will not address the thornier problems that arise in connection with generic reference and non-specific indefinite reference. The particular anaphoric devices we will explore are pronominalization, definite noun phrase anaphora, and *one*-anaphora. It will be useful to set our goals in these areas in the context provided by the other systems we have discussed.

2.6.1 Pronominalization

Given our interest in generating connected discourse, our primary focus with respect to pronominalization will be the generation of intersentential pronominal anaphora: we will have little to say about intrasentential anaphora, and in particular, we will sidestep the issues that arise in dealing with bound variable anaphora.

Of the approaches to the generation of pronouns we saw in the previous section, the

most elaborate was that offered by McDonald [1980a]. Of course, the most elaborate is not necessarily the best. Indeed, in a footnote, McDonald suggests that, rather than use the battery of heuristics employed by MUMBLE, a notion of discourse focus might be sufficient to decide upon pronominalization [McDonald 1980a:220]. The particular approach we adopt in the present work (described in chapter 5) is closer in spirit to the method used by Davey [1978], although it makes use of the intuitions present in the centering theory of Sidner [1979] and Grosz, Joshi and Weinstein [1983]. In conjunction with some observations we make about long distance pronominalization, this permits the use of a very simple approach to pronominalization which does not require the 'second-guessing' of the hearer's interpretation of a pronoun used in MUMBLE and HAM-ANS to ensure that a pronominal reference will be satisfactory. It remains to be seen, however, to what extent this simple approach can be used in domains other than that used in the present work.

2.6.2 Definite Noun Phrase Anaphora

All the systems described adopt broadly similar approaches to the generation of definite noun phrase anaphora: that is, they see it as a process of constructing a description which serves to distinguish an intended referent from a set of discourse entities. However, very little detail is specified as to the algorithms used to carry this out. This is one area we will examine more closely in chapter 5.

In addition, it is noticeable that the systems we have described are all concerned with relatively simple entities. In chapter 3, we examine the kinds of entities that occur in cookery recipes in some detail, and arrive at a representation for these entities that is considerably more complex than those used by the systems discussed in the previous section. This has consequences for the algorithms used to generate referring expressions, but also permits us to generate a wider range of noun phrase structures, as we will see in chapter 5.

Finally, as we saw in section 2.3, the question of whether or not a definite determiner can be used is far from simple. Other than the fairly simple model used in Davey's PROTEUS, none of the systems we have seen have addressed this question directly. In chapter 5, we describe how this phenomenon might be dealt with in the framework

presented here, using a mechanism which is essentially an extension of that suggested by Davey.

2.6.3 One-anaphora

Of the generation systems we have examined, only Davey's PROTEUS and HAM-ANS address the problems that arise in the generation of *one*-anaphora in any depth. Although there are useful ideas in Davey's approach, his use of a tightly restricted domain limits the general usefulness of his algorithm. The approach we take is closer to that used in HAM-ANS in that it involves checking for the possibility of ellipsis at the level of semantic structure; however, we incorporate some of the insights offered by Webber [1979], using two levels of semantic structure to allow us to deal with a wider range of *one*-anaphoric expressions.

Chapter 3

The Representation of Entities

In the previous chapter, we looked at the wide range of referring and anaphoric forms available in natural language. In this chapter, we consider the other end of the reference relation, focusing our attention on the kinds of referents that occur in a particular domain: that of cookery. As noted in chapter 1, cookery recipes are an ideal testbed for the issues addressed in this thesis, since they require a complex ontology, and allow us to make use of the notion of generating text from plans. The aim of the present chapter is to develop an ontology that encompasses at least the more common of the range of entities that can be referred to within that domain: our major concern is the representation of complex entities in such a way as to allow us to model their behaviour and change of state as the execution of a recipe proceeds, while at the same time making it straightforward to construct references to those entities.

We take the view that there are two kinds of entities in the world: informally, *things* and *events*. We are first and foremost concerned with the representation of things, although some use will be made of the parallels that can be drawn between things and events, and to that extent the representation of events will also be discussed.

In section 3.1, we discuss some of the basic issues that arise from a consideration of what constitutes an object in a recipe.

In section 3.2, we provide a more detailed survey of the variety of referring expressions that occur in recipes: this gives us a better idea of the kinds of phenomena our ontology, and therefore our representation of the world, has to deal with.

In section 3.3, we introduce the notion of a GENERALIZED PHYSICAL OBJECT or PHYSOBJ, and describe how this can be used as the basis of a representation that can handle at least a considerable subset of the data previously considered. The mechanisms adopted allow us to deal with both the mass/count distinction and plural objects in a general way.

In section 3.4, we introduce the other basic kind of entity we permit in our ontology, the EVENTUALITY. The treatment of eventualities has strong parallels with the approach we take to objects, although it is dealt with somewhat less exhaustively, given our primary interest in references to objects. However, sufficient detail is laid out to connect the basic notion of an event with the notion of a plan of action used in chapter 4 as the basis for the generation process.

In section 3.5, the representational language is summarised; and section 3.6 ends the chapter by indicating some known limitations of the approach taken here, and some suggestions for solving the associated problems.

3.1 Some Ontological Problems in Recipes

Ontology is not easy, particularly in a domain as complex as cooking. The principal task carried out by EPICURE is the construction of referring expressions that pick out objects in the domain; this requires that each object of interest in the domain be represented by a distinct symbol. From the point of view of representing the ingredients in a specific recipe, the first problem we have to face is that of deciding what constitutes an object.

3.1.1 The Limitations of Singular Individuals

In most natural language systems, it is assumed that all the entities in the domain of discourse are singular individuals. Typically, each such entity is represented by a symbolic constant, which serves as a locus of information: the properties of the entity can then be represented by asserting propositions that predicate arbitrary properties of that entity. One of the most pervasive of the properties so expressed is what we might call *sortal class membership*, and is represented by the ISA link found in most knowledge representation formalisms (although not always under that name): so, for

example, if we have symbolic constants e_1 and e_2 that correspond to a table and a chair respectively, this information might be represented as follows:

(3-1) `(#IS :E1 #TABLE)`
`(#IS :E2 #CHAIR)`

where the #IS concept is taken to mean something like 'has as its basic description' [Winograd 1972:118]. A number of variations on this basic syntax are used to represent the same information; thus, in Appelt's system [Appelt 1985:132], we have the simplified forms

(3-2) `(Table (E1))`
`(Chair (E2))`

In each case, the representation language embraces some finite set of object categories, and each object in the domain of discourse is said to be an instance of one of these categories.

However, as we consider more complex domains such as recipes, we find that this simple notion is of limited value. Of course, we do find ingredients such as

(3-3) a carrot

and it would seem reasonable in such cases to represent this, analogously to the examples above, as something like

(3-4) `(isa x carrot)`

where x is the internal symbol corresponding to the carrot in question. However, simple objects like these are the exception in the world of cooking. We find, for example, ingredients like

(3-5) three carrots

In such cases, we might take the view that we have a single noun phrase which specifies some plural number of singular individuals collected together. This is still possible within the representational framework used above: we could represent the set of objects in question using the following conjunction of expressions

(3-6) `(isa x1 carrot)` $\wedge$ `(isa x2 carrot)` $\wedge$ `(isa x3 carrot)`

and then invoke some mechanism to collect together these three objects in order to produce an appropriate referring expression. Plurality is then an issue for the generator, but not for the representation language.

This approach will work as long as we have an internal symbol for each individual that figures in the collection of individuals described. However, this is not always the case: we also find examples where the cardinality of the set described by the noun phrase is not specified, as in

(3-7) three pounds of carrots

In these cases, the meaning of the noun phrase cannot be represented at all if we only permit ourselves singular individuals, since we do not know how many individual carrots the set in question comprises; intuitively, we want to be able to say that the noun phrase refers to a *set* of singular individuals, and so we need to be able to represent such an entity using our formalism.

Other linguistic forms which cause related problems are those that make use of mass nouns rather than count nouns. The objects described are usually specified in terms of the substance of which they consist, in conjunction with some measure of quantity: for example

(3-8) 3 tablespoons of water

Here we have neither a set nor a singular individual, and so we have to allow for a third kind of object that can be referred to by a noun phrase.

Of course, none of this is controversial from the point of view of linguistics: we know that nouns can be count or mass, and that noun phrases incorporating count nouns can be singular or plural. However, although semanticists have examined these problems at length (see, for example, the collection edited by Pelletier [1979], and more recent work by Link [1983]), and some work in knowledge representation within AI addresses related issues (in particular, Hayes [1974, 1978, 1985]), there seems to be no work in computational linguistics that works out the ramifications of adopting a corresponding ontology.

3.1.2 The Individuation of Ingredients

A related problem to that just discussed is the question of deciding exactly how many distinct objects we are dealing with in a given situation. Suppose we have the following list of ingredients:

(3-9) one egg
$\frac{1}{2}$ pint milk
4 oz cheese
salt

It seems reasonable to suggest that this list specifies four ingredients. However, suppose the list specifies two eggs rather than one:

(3-10) two eggs
$\frac{1}{2}$ pint milk
4 oz cheese
salt

Do we now have five ingredients in total, or are there still only four?

There are a number of possible approaches to this. We might take the view that the answer depends on the recipe itself: so, in the above example, we might say that if the eggs are used together, then they constitute one ingredient, whereas if they are used separately, they constitute two ingredients. Under this view, in

(3-11) Beat the eggs ...

they serve as one ingredient, but in

(3-12) Beat one of the eggs ...
Use the white of the second egg to ...

the two eggs serve as distinct ingredients, and the fact that both are specified in the same item of the ingredients list can be attributed to some consideration of style or convenience (for example, listing them together makes it easier to determine what you might have to buy at the shop).

However, this presumes a clear understanding of what it means to be 'used together'. Suppose we have an ingredients list containing the following items:

(3-13) $\frac{1}{2}$ pt milk
$\frac{1}{2}$ pt vegetable stock

and the following instruction in the recipe:

(3-14) Now add the milk and stock ...

There is a clear sense here in which the milk and stock are being used together, and yet we would not want to say that they together constitute one ingredient.

Another approach would be to assume that each separate *orthographic* item in the ingredients list is a distinct ingredient. This too is problematic: in some cookery books, for example, salt and pepper appear in ingredients lists as separate items:

(3-15) salt
pepper

whereas in others, they are listed together:

(3-16) salt and pepper

Are we to say that, in one case, we have two ingredients but that in the other we have only one? It seems more plausible to say that different books follow different conventions for *structuring* the ingredients list. Thus, in some books we find lists like the following:

(3-17) one onion
two potatoes
one carrot

but in others we find

(3-18) one onion, two potatoes and one carrot

This does not immediately help in deciding how many distinct ingredients we have; however, it is compatible with yet another approach, where we might stipulate that each distinct noun phrase appearing in an ingredients list describes a distinct ingredient, even if that noun phrase is a conjunct within a larger noun phrase. To put this another way, if something warrants a distinct description, then it ought to be considered a distinct object. Under this view,

(3-19) two eggs

describes one ingredient, and

(3-20) one onion, two potatoes and one carrot

describes three ingredients. In the latter case, the way in which these ingredients are then collected together for description is the generator's responsibility: there is no symbol corresponding to the conjunction of the onion, potatoes and carrot. By providing a parameter to control the style of output, either of the above orthographies could be generated.

However, what are we then to say of ingredients like the following?

(3-21) 450g mixed vegetables: carrot, cut into rings; potato, swede, diced; celery, leeks, sliced; cauliflower, broken into largish sprigs

The approach just suggested would require us to view this as a list of six ingredients (the carrot, potato, swede, celery, leeks and cauliflower). The difficulty here is that we know the weight of the six ingredients taken together, but not separately, which presents us with a problem when it comes to representing the weight; ideally, we have to provide the system with an additional symbol corresponding to the 'higher level' ingredient, so that we have somewhere to attach the weight information.

In this thesis, we take the view that language itself is the best guide to ontology. As a result, we take each distinct noun phrase to correspond to an object, and we permit objects to contain other objects as constituents, so that the ingredients list input to the generator may already contain some prestructuring. Thus, in

(3-22) salt and pepper

we have an object which has two objects (one a quantity of salt and the other a quantity of pepper) as constituents; and in

(3-23) 450g mixed vegetables: carrot, cut into rings; potato, swede, diced; celery, leeks, sliced, cauliflower, broken into largish sprigs

we have an object (the *mixed vegetables*) which has six objects as constituents. On the other hand, in

(3-24) two eggs

we have a single object. This does not preclude the possibility that subsequently in a recipe we might wish to decompose this object to produce two distinct objects which

are referred to separately: however, we only carry out such proliferation of objects when this is necessary in order to be able to represent the required information.

3.2 A Survey of Objects in Recipes

Now that we have a better idea as to what we take to be an object within the cookery domain, we can go on to examine the wide variety of referring expressions to be found in recipes in more detail. In particular, we consider the referring expressions that appear in the ingredients lists at the beginning of recipes.[1] A typical ingredients list might look like the following:

(3-25) 225g roasted buckwheat
300ml water
2 teaspoons yeast extract
1 tablespoon tomato purée
2 large onions, chopped
2 sticks celery, diced
1 tablespoon olive oil
4 garlic cloves, crushed
350g dark, open mushrooms, washed and sliced
salt, pepper and tamari

Even within this relatively simple example, closer inspection reveals a great many complexities which are not immediately apparent. These complexities will be discussed in what follows. In this section, a range of data is presented in order to exemplify the variety of expressions that are to be found in real recipes. The representation language presented in the next section will not be able to represent the meanings underlying *all* of the forms presented here, although it is capable of handling a considerable proportion of them, and all of the more common ones.

We can view each ingredient listed at the beginning of a recipe as a specification of one object that plays a particular *role* in that recipe. The functional role of an ingredient within a recipe is its primary characteristic: the question of which particular substance or substances is used to fulfil this role is often relatively unimportant, and so an experienced cook can often replace the specified ingredient by substituting some other ingredient that is capable of performing the same role. Thus, when making a curry,

[1] The data that follows are drawn from two particular cookery books (*Beanfeast* by Rose Elliot, Fontana 1985, and *The Vegetarian Epicure* by Anna Thomas, Penguin 1973) but are representative of those found in many others.

if you have no ghee it is quite acceptable to replace the ghee by some other substance suitable as a base for frying, such as oil or butter; when making green lentil dahl, the green lentils themselves could be replaced by some other kind of lentils without disastrous consequences (although it would then be inappropriate to call the end result 'green lentil dahl').

There is much to be said for representing the ingredients in recipes in terms of their functional roles, as we shall discuss later; however, this is a considerable enterprise in its own right, and so in the current work we will continue to talk of ingredients solely in terms of their physical instantiations. Thus, each ingredient in an ingredients list is taken as specifying one preferred physical instantiation of the role to be filled by that ingredient.

As suggested in the previous section, our first problems come with the view that objects in the domain can be treated as singular individuals.

3.2.1 Conjoined and Disjoined Specifications

If more than one ingredient performs a particular role in a recipe, these ingredients will typically be listed together in a conjoined noun phrase as a single entry in the ingredients list: thus we have

(3-26) a salt and pepper
b chives and parsley, chopped

where in the first case the two ingredients together typically function in the role of 'seasoning', and in the second case the two ingredients serve as a garnish. It is not only such 'minor' ingredients that can be grouped together in this way: for example, in a recipe for a fruit-based dish, we find

(3-27) 2 apples, 2 pears, 1 orange, 1 banana

appearing as a single entry on the ingredients list. Notice that, in conjunctions of this sort, the descriptions applied to individual conjuncts take account of the context provided by the superordinate noun phrase:

(3-28) 4 whole eggs and 3 egg whites

This would sound odd as

(3-29) 4 eggs and 3 egg whites

Of course, context-sensitive description also occurs within the ingredients list as a whole, so that we find forms like

(3-30) a extra olive oil or ghee for frying
b more fresh sliced pimento

used when the recipe already contains these substances for other purposes.

The conjunction of ingredient specifications may also be achieved by quantifying across a number of separate ingredients, as in

(3-31) a 10 each of allspice berries, juniper berries and black peppercorns
b $\frac{1}{2}$ teaspoon each of allspice, cinnamon and grated nutmeg

At the very least, phenomena such as these argue for the use of something more complex than a simple flat-structured list of ingredients in the representation of a recipe. As we have already seen, sometimes an item in the ingredients list will itself be a list of objects along with a top level description:

(3-32) a 1 cup mixed vegetables: carrot, cabbage, celery, tomato
b raw vegetables: radishes, spring onions, carrot sticks, pieces of celery and cauliflower florets

In such cases, the list need not be exhaustive, but may provide examples, as in the following:

(3-33) a 1 teaspoon dried herbs: peppermint, sage or rosemary, for instance
b 1 tablespoon chopped fresh herbs, such as mint or coriander

The specification of an ingredient can also involve disjunction, offering alternative physical instantiations for a particular role within a recipe. Thus we have:

(3-34) a 1 tablespoon rolled oats or 1 dessertspoon medium oatmeal
b 1 very large potato or 2 small potatoes
c 75g chopped almonds or hazelnuts

Just as in the case of conjunctions, this list of alternatives can be provided with a higher level description:

(3-35) 550ml fruit juice: pineapple, apple, orange or grape

Note that disjunctions and conjunctions can be combined within the same ingredient specification, as in

(3-36) 2 small pots yoghurt and 0.25 pint water or 0.75 pint buttermilk

The mechanisms used in this thesis are adequate for the generation of most conjoined specifications. However, we will not attempt to generate disjunctive specifications, despite the fact that these are surprisingly common in recipes. The problems associated with disjunctive specifications of ingredients are discussed further at the end of this chapter.

3.2.2 Ingredient Taxonomies and Object Packaging

We saw above that it is not sufficient to posit straightforward correspondences between object instances and 'types' such as *carrot*. However, even before we deal with the problems arising from allowing mass and plural objects, there are other difficulties connected with the allocation of objects to categories. Not all ingredients have simple names such as *carrot* and *potato*: often, names are compound, as in the following:

(3-37) a garlic clove
b alfalfa sprouts
c eating apple
d avocado skin

Note that there are various ways in which these names can be derived: a garlic clove is a clove-shaped *package* of garlic-matter, whereas an alfalfa sprout is a particular *kind* of sprout. In this respect, an eating apple is analogous to an alfalfa sprout; an avocado skin, however, is *part of* an avocado. The representation language to be presented later in this chapter is capable of making these distinctions.

This is particularly important from the point of view of subsequent reference: for example, a garlic clove is more likely to be subsequently referred to as *the garlic* than as *the clove*, whereas an eating apple can be subsequently referred to as *the apple*, but not as *the eating*. This suggests that the generation of subsequent descriptions depends on a *semantic* issue, that is, what substance is described, rather than a *syntactic* issue, that

is, whatever the head of the antecedent noun phrase was: quite a different mechanism to that used in most existing language generation systems, a point we will pursue in more detail in chapter 5. Note, however, that any subsequent references to an *avocado skin* are likely to be of the form

(3-38) the skin

This suggests that, in the case of objects which are parts of other objects, the part name is the important element from the point of view of subsequent reference.

3.2.3 Quantities

In general, the description of an ingredient consists of a specification of two things: its substance and its quantity. The quantity of an ingredient can be specified in several ways. As we saw above, sets of singular individuals can be specified in terms of the cardinality of the set, as in:

(3-39) three carrots

Quantities of mass substances, on the other hand, are specified by means of volume or weight:

(3-40) a three pints of milk
b two pounds of rice

Sets of individuals can also be specified by weight:

(3-41) three pounds of carrots

Apart from the standard metrics for solids and liquids, quantities can also be specified using measuring devices such as *teaspoons* and *cups*, as in

(3-42) 1 cup of olives

These sorts of measurements may also be qualified, as in

(3-43) 1 heaped tablespoon tahini

Quantities can be specified to varying degrees of exactness: a measurement can be

approximate as in

(3-44) about 1.5lb unbleached, hardwheat, white flour

or it can be specified as a range as in

(3-45) 1 to 2 slices dark bread

There are also relatively vague units of measurement as in the following:

(3-46) a a bunch of watercress
b a handful of raisins
c a dash of sugar
d a drop of tabasco
e a little butter
f a few drops of wine vinegar

The cardinality of a countable set can also be left imprecise, as in

(3-47) a a few chopped nuts
b a few lettuce leaves

Finally, in some cases, there is no specification of quantity at all, as in the following:

(3-48) a honey
b sea salt
c chopped fresh herbs

The quantity of a compound ingredient can be specific while the amounts of the component ingredients can be left unspecified:

(3-49) 350g mixed raisins, sultanas and currants

All of the above, with the exception of approximate measurements, will be catered for by our representation language.

3.2.4 Properties

The specification of an ingredient often includes properties of the ingredient other than the substance it consists of. As is to be expected, the properties mentioned are just those that are relevant from the point of view of cooking. Some of these properties are

best thought of as properties of the way the ingredient is 'packaged', whereas others are best thought of as properties of the substance itself: so, for example, we might be told that something is *large* or that something is *ripe*. Some properties of objects can be used to derive names of subtypes: for example, we talk of *button mushrooms* as a special type of mushroom, where the name is derived from the shape of the mushroom. Thus, some properties would seem to license the construction of a type hierarchy for ingredients, whereas this seems to be less the case with other properties: *1 large eating apple* sounds like a large instance of the specialization of apple known as an eating apple; on the other hand, we might be less happy about saying that a large onion is a specialization of onion. This distinction can be catered for by means of the distinction between properties of packagings and properties of substances just described.

On occasion, we find that negative properties are specified:

(3-50) a 225g cooking dates (not 'sugar-rolled')
b 1.5 oz currants (not raisins)

A reason for the negative specification may be offered:

(3-51) 125g light muscovado sugar (not the very dark one, as it's too treacly for this)

We will not, however, deal with properties specified by means of negatives in this thesis.

3.2.5 States and Processes

Very often, some properties used in the specification of ingredients are properties arising from some process having been applied to the ingredient: so, for example, we have

(3-52) pitted ripe olives

where the property of being pitted arises from a process of *pitting*. In such cases, the author of the recipe may intend that the reader carries out this processing, or that the ingredient be obtained already in this state. Sometimes one or other is clear from what we know about ingredients: for example, at the time of writing, it is not possible to buy onions which have already been both peeled and chopped, so we would assume that an ingredient specified as

(3-53) 2 peeled and chopped onions

is intended to be taken by the reader as an instruction to peel and chop the onions. On other occasions, however, the description is ambiguous precisely because, as in the case of the *pitted ripe olives* above, the ingredient can be obtained both processed and unprocessed. The important thing, then, is that the ingredient should have the specified properties before the instructions in the recipe proper are carried out.

Sometimes the specification of preprocessing can be quite complex. So, for example, operations that are to be carried out on objects can be specified more precisely by means of adverbial modification as in

(3-54) freshly ground black pepper

where it is important that not only is the pepper to be ground, but it must have been ground very recently; similarly,

(3-55) 450g tender spinach, washed and finely shredded

must have been shredded *finely*. Things do get more complex still, however: consider

(3-56)
a 1 cauliflower, washed and broken into florets
b 2 medium-sized aubergines sliced into 6mm circles
c 225g dried unsulphured apricots, washed and covered with boiling water, soaked overnight
d 2 oz wheat germ, toasted with honey
e 3 cloves garlic, put through a press

In these cases, processes as complex as some of those that appear within the body of the recipe itself appear to have migrated into the ingredients list. The specification of processing can also occur in conjunction with the kinds of structures noted above in our discussion of conjunctions and disjunctions:

(3-57) 450g mixed vegetables: carrot, cut into rings; potato, swede, diced; celery, leeks, sliced; cauliflower, broken into largish sprigs

Here, what could have been an entire paragraph in a recipe finds itself realized as a noun phrase in the ingredients list.

One particularly troublesome aspect of the incorporation of processing information into

a description has to do with the time at which particular forms of description are appropriate: for example, if we have

(3-58) a carrot

which is subsequently grated, the result of this operation is best described as

(3-59) some carrot

However, it is not uncommon to find descriptions like

(3-60) one carrot, grated

in a recipe, where the first part of the noun phrase (*one carrot*) describes the object in question *before* the processing described in the second part (*grated*) has been applied. Thus, describing an object may require knowing what is true of that object at different times.

Our representation language is capable of representing most of the above kinds of processing information.

3.2.6 Derived Objects

Above, we saw that objects can be described in part by describing the processing that has been applied to them. A closely related phenomenon is that ingredients can be introduced as parts of other objects from which they are derived, as in

(3-61) a slice of lemon

These 'derived objects' can be described in a number of different ways: for example, as being of a particular shape, as in the previous example; or as some proportion of the 'ancestor' object, as in

(3-62) half a carrot

Alternatively, the object may be described as a structural part of another object, as in each of the following:

(3-63) a an apple core
b an egg yolk
c the kernels from 1 fresh ear of corn
d seeds from 2 to 3 cardomom pods

Or again, the object may be described in terms of the process used to derive the object, as in

(3-64) a a squeeze of lemon juice
b peels from 6 to 7 large, healthy potatoes

Notice that processing can be specified for either the ancestor object or the derived object:

(3-65) a grated orange rind,
b grated rind of a well-scrubbed orange
c grated rind and juice of 1 small well-scrubbed orange

Disjunctions can also enter into specifications like this:

(3-66) grated rind of 1 orange or lemon

Note also that properties of both the derived object and the ancestor object may be specified, as in

(3-67) half a fresh pimento, thinly sliced

where it is the pimento that is fresh, but the half-pimento that is thinly sliced.

Ingredients can also be specified as the *omission* of some part of something else:

(3-68) a 175g wholewheat bread without crusts
b 225g strawberries, stalks removed

The mechanisms described in this thesis are capable of representing a wide range of derived objects, although some of the more complex cases are not dealt with.

3.2.7 Miscellaneous Other Observations

There are a number of other less frequently occurring phenomena to be found in the specification of ingredients. These are mentioned here for completeness, although none

of the following are addressed by the representational mechanisms described in the rest of this chapter.

The specification of an ingredient sometimes states the function of that ingredient within the recipe:

(3-69) a lemon rings to garnish
b lettuce leaves to serve

Statement of purpose in this way is relatively rare, and for the most part is restricted to those ingredients which are in some way peripheral to the dish under construction; there are exceptions to this, however, as in

(3-70) olive oil or ghee for frying

A notion of functional role, as mentioned at the beginning of this section, would be helpful here.

There are many other 'adjuncts' to ingredient specifications. Sometimes an ingredient may be specified as optional, as in

(3-71) a little honey to taste, optional

Some optional ingredients specify the condition that has to be satisfied in order for them to be included:

(3-72) a low-sodium yeast extract, if liked
b 1 teaspoon chopped fresh rosemary, thyme or marjoram if available

Sometimes the source of an ingredient may be specified as a modifier:

(3-73) 450g dried fruit salad mix, from health shops

Finally, some ingredients cross-refer to other recipes, as in

(3-74) 1.5 pints aubergine pasta sauce (page 257)

All of the above are sufficiently rare that we can ignore them in what follows without thereby imposing any serious limitations on the coverage of the representation language.

3.3 Generalized Physical Objects

In this and the following sections, we present a representation language that encompasses a wide range, although not all, of the examples discussed in the previous section. The ontology encompasses two basic types: eventualities (corresponding to states, actions and events), and generalized physical objects (corresponding to individuals, masses, and sets of individuals). The focus in this section is on physical objects; in section 3.4 we consider eventualities.

First, in order to be able to represent both singular, plural and mass objects, we introduce the notion of a GENERALIZED PHYSICAL OBJECT. Then, after some discussion of the count/mass distinction, we introduce the notion of an object's STRUCTURE as a way of categorizing it as singular, plural or mass. We then go on to describe the representations used for a variety of kinds of objects, concluding with the representation of simple properties. The representation of derived objects and objects which have a complex processing history is left until we have described the representation of eventualities in section 3.4.

3.3.1 The Generalized Physical Object

As we have now repeatedly emphasised, it is insufficient to assume that all we need in order to represent the kinds of objects that occur in recipes is the notion of a singular individual. Nonetheless, each item in an ingredients list (with some exceptions to be discussed below) does appear to function as a distinct object in a recipe, irrespective of how that ingredient is composed; thus, if we have an ingredient introduced as

(3-75) 3lbs of carrots

it is relatively rarely that some *part* of that ingredient will be referred to: the carrots are typically used subsequently in the recipe as if, at least from the point of view of describing them, they constitute an indivisible object.

In the light of the earlier discussion concerning the individuation of objects in recipes, the ontology developed here echoes a suggestion made by Hobbs: that we should multiply the kinds of entities we allow in our ontology *by allowing as an entity everything that*

can be referred to by a noun phrase [Hobbs 1985:61]. Complications arise in taking this view to its extremes: for example, it would require that, in the sentence

(3-76) Either Michael or Mary will feed the cat.

we posit the existence of a disjunctive entity corresponding to *either Michael or Mary*. For simplicity, we ignore this possibility in the present work, and consider only specific individuals which are described by singular, plural and mass terms: as stated earlier, there are problems in dealing with disjunctions, a matter we will return to at the end of this chapter.

We view each ingredient in a recipe as a GENERALIZED PHYSICAL OBJECT, or, for short, a PHYSOBJ. A physobj is defined as follows:

(3-77) A generalized physical object is any (not necessarily contiguous) collection of contiguous regions of space occupied by matter.

Thus, a physobj consists of one or more 'conventional' physical objects. It is convenient to think of a physobj as a collection of physical objects brought together for the purpose of referring to them (although a physobj does not need to be referred to in order to exist). Thus, entities that exist because they consist of molecules brought together in the nature of the world (e.g., the referent of *an onion*), and entities brought together for the purpose of referring to them by means of noun phrases (e.g., the referent of *two onions*) are physobjs of equal standing.

Given the above, each of the following noun phrases has as its referent a physobj:

(3-78) a an onion
b two pounds of rice
c Michael and Mary

There is nothing in the definition of a physobj which requires that the constituent physical objects be made of the same substance, or that they should be connected or in each others' proximity: a physobj is just some collection of matter which we find it convenient to view as an individual.

Note that, although the notion of a physobj is more general than that of an ordinary physical object, it is not as general as Hobbs' more promiscuous notion of an entity mentioned earlier. Quite apart from the question of how we deal with disjunctions, it

should be noted that Hobbs' notion of an entity does not require that the entity have any sort of physical existence: in contrast, we do not consider abstract objects here.

By the above definition, the noun phrase *Jon and William* describes a single physobj irrespective of whether Jon and William are near each other; so, *Jon and William* describes an object in both of the following:

(3-79) a Jon and William went to the supermarket together.
b Jon and William have not yet met.

Ultimately, we may not require every NP to specify an object in this way; but the notion of a physobj provides a framework where it is possible.[2] We can later restrict this possibility if this seems desirable.

3.3.2 The Count/Mass Distinction

As we noted earlier, some objects are described by *count* noun phrases, and some by *mass* noun phrases:[3]

(3-80) a a large carrot
b some grated carrot

It is tempting to suggest that there is an ontological distinction underlying this linguistic one, and indeed this would seem to be borne out in the general case: mass noun phrases tend to be used to describe gases, liquids, powders and collections of small objects (such as grains of rice) which are sufficiently small for it to be inconvenient to view them as individuals in their own right; count noun phrases, on the other hand, are used to describe objects which have identifiable boundaries. However, positing an ontological basis for the count/mass distinction is not without its problems. Lyons [1968:281-282], for example, argues that the count/mass distinction is primarily a linguistic one, evidence for this being drawn from cross-linguistic data: some substances are denoted by count nouns in one language but by mass nouns in another. Thus, the English word *grape* is a count noun, whereas the German *Traube* and the Russian *vinograd* are mass terms [Lyons 1968:282].

[2] Again, with the exception of disjunctive objects.

[3] In the literature, the terms 'count' and 'mass' tend to be predicated of nouns rather than noun phrases; however, by extension, we can say that a count noun phrase is one which has a count noun as its head, and a mass noun phrase is one which has a mass noun as its head.

Another problem is that there are borderline cases where there appears to be a conflict between the naive ontology and the linguistic facts. Speaking ontologically, there appears to be little difference between individual rice grains and individual lentils: although they consist of different substances, they are both small objects of roughly the same size. However, when describing quantities of these, we use a mass noun in the first case but a count noun in the second case:

(3-81) a four ounces of rice
b four ounces of lentils

If the count/mass distinction was ontologically based, we would expect these descriptions to be either both count or both mass.

It is more plausible to suggest that what we require is a *conceptual* distinction: i.e., we want to encode whether or not something is *conceived of as* count or mass. Leech and Svartvik take a similar view:

> Some mass nouns, we might argue, should 'really' be count, because the 'substance' is divisible into separate things ... But PSYCHOLOGICALLY we think of such things as indivisible when we use a mass noun.
>
> [Leech and Svartvik 1975:45]

We take the view here that physobjs are not inherently count or mass, but are *viewed* as being count or mass: we say that a physobj may be viewed from either a mass or count PERSPECTIVE. Thus, a specific physobj can be viewed at one time as a mass, and at another time as a countable object: when cooking, I will in all likelihood view a quantity of rice as a mass, but if I am a scientist examining rice grains for evidence of pesticide use, I may view that same quantity of rice as a countable set of individuals. In the domain of cooking, there is very rarely any need to use or maintain more than one perspective on a given object, and so for purposes of the system described in this thesis, we can view the countability or otherwise of an object as if it were an objective property of that object; however, at the end of this chapter we will consider some circumstances where this approach is inadequate.

3.3.3 Representing Quantities of Stuff

Each object in the domain is represented by means of a KNOWLEDGE BASE ENTITY, or KB entity for short. For the purposes of exposition, first order predicate calculus (FOPC) will be used in what follows as a formalism for representing the information contained in KB entities; however, the system described in this thesis makes use of FEATURE STRUCTURES as its core representational mechanism. In some cases below, both the FOPC form and the corresponding feature structure will be shown in order to demonstrate the relationships between the two. The feature structure language is specified more formally in section 3.5.

Objects

In the real world, we might take the view that any two entities which have exactly the same properties are, in fact, the same entity. However, in the mental world—where incomplete information about an object is the rule rather than the exception—it is entirely possible that two distinct entities may share all the same properties. For each entity in the domain, then, we introduce a unique symbol, which serves as a locus of information that describes that entity. This symbol is the entity's INDEX. Properties of the entity are predicated of this symbol. The most important property of an entity is its basic ontological type: every object is either a physobj or an eventuality. For convenience, we use sorted indices in the work described here: the indices x, y, z and subscripted versions thereof should be read as representing physobjs, and the indices e, f, g and subscripted versions thereof should be read as representing eventualities (introduced in section 3.4). Every ingredient in a recipe is a physobj, and is therefore represented by a symbol of the form x_i.

Irrespective of whether an ingredient is described by means of a count or a mass expression, it consists of some QUANTITY of some SUBSTANCE (although in some circumstances the quantity may not be specified). These properties of the object, which are akin to Link's INVARIANT PROPERTIES [Link 1983:304–305], are unchanging throughout the life of the object, whereas any other properties of the object may change. This simplifies the process of generating subsequent referring expressions: in general, it is very easy to determine if we are talking about the same entity as was mentioned on a specific previ-

ous occasion. However, this approach leads to complications when dealing with entities described by noun phrases such as *the soup*, where the constituency of the object, and therefore the collection of substances of which it consists, changes during the process of the soup being cooked.

Substances

The substance of an object is specified as one of a set of constants which form a hierarchy of substances: thus, an ingredient which is a quantity of low-fat milk has its substance specified as

(3-82) substance(x, low-fat-milk)

where we specify separately that low fat milk is a kind of milk, and that milk is a kind of liquid:

(3-83) ako(low-fat-milk, milk)
ako(milk, liquid)

As noted above, entities are represented within the system as feature structures. The KB entity representing the information in (3-82) is then

(3-84)
$$\left[\begin{array}{l} index = x \\ spec = \left[\ substance = low\text{-}fat\text{-}milk\ \right] \end{array}\right]$$

In the feature structure representation, all properties predicated of an entity are attribute-value pairs within the value of the entity's SPEC (i.e., *specification*) attribute.

Quantities

The quantity of an ingredient may be specified in a number of ways, typically by weight or volume. A weight or volume is specified as a tuple consisting of a unit of measurement and a number which specifies how many units of that type there are. Thus, the quantity and substance of the ingredient described by

(3-85) 4 oz cheese

is represented as

(3-86) $\exists x$ substance(x, cheese) $\wedge$ quantity(x, <4, ounce>)

We permit a wide range of units of measure that can be used to specify the quantity of an ingredient: we make no syntactic distinction between standard units of measurement such as *ounces*, *grammes*, *litres*, and *pints*, and more informal measures such as *teaspoons*, *cups* or even *handfuls* and *dashes* (although lexical information will determine whether, for example, we should say *one teaspoon* but *a dash*). So, for example, we represent the quantity of substance described by

(3-87) a dash of tabasco

as

(3-88) $\exists x$ substance(x, tabasco) $\wedge$ quantity(x, <1, dash>)

We can also specify the quantity of an ingredient as a RANGE, where a range consists of an upper limit and a lower limit; each is expressed as a quantity tuple. The quantity of substance described by

(3-89) 1–2 tablespoons honey

is therefore represented as

(3-90) $\exists x$ substance(x, honey) $\wedge$
quantity(x, range(<1, tablespoon>, <2, tablespoon>))

For ease of manipulation, the feature structure representation of this information is slightly more structured: the feature structure corresponding to (3-90) is then

(3-91)
$$\left[\begin{array}{l} \mathit{index} = x \\ \mathit{spec} = \left[\begin{array}{l} \mathit{substance} = \mathit{honey} \\ \mathit{quantity} = \left[\begin{array}{l} \mathit{lowerlimit} = \left[\begin{array}{l} \mathit{unit} = \mathit{tablespoon} \\ \mathit{number} = 1 \end{array}\right] \\ \mathit{upperlimit} = \left[\begin{array}{l} \mathit{unit} = \mathit{tablespoon} \\ \mathit{number} = 2 \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

Note that the index of an entity is essentially *extensional*, whereas the specification is

essentially *intensional*. There can be many entities with the same specification, but no two entities can have the same index. The specification of an entity can thus be viewed as a concept or type.

3.3.4 States

Apart from the constant properties described in the previous section, an object may also have properties which are only true of that object at certain times. We say that a property is true of a given object in a specific STATE, where a state is a partial description of the world at some point in time. A state can be characterised by a formula

(3-92) $P_1 \wedge P_2 \wedge \ldots \wedge P_n$

where each P_i is a proposition that is true in that state. Rather than represent states as long conjunctions of propositions, we represent them in a piecemeal way by stating that some proposition HOLDS in a given state: thus, if we know that an object x is hot in the state s, we say

(3-93) holds(s, hot(x, +))

Thus, the characterisation of a state presented above is equivalent to the following:

(3-94) holds(s, P_1) $\wedge$ holds(s, P_2) $\wedge \ldots \wedge$ holds(s, P_3)

supposing s to be the state described.

States are temporally ordered; thus, assuming a function *time* which maps a state into the temporal location it describes, the following relationship holds for any two states s_i and s_j:

(3-95) [time(s_i) < time(s_j)] $\vee$ [time(s_i) = time(s_j)] $\vee$ [time(s_i) > time(s_j)]

The invariant properties described in the previous section hold true of the objects of which they are predicated in all states:[4] thus, a more correct representation of the quantity of substance described by

[4] In reality, of course, objects are not eternal, even if the particles they consist of are. However, it is convenient to view ingredients as existing for the duration of a recipe, even if they lose their status as distinct objects, and so we can take s to range over the set of states that occur during the execution of a recipe.

(3-96) 4 oz cheese

is

(3-97) $\exists x \forall s$ holds(s, substance(x, cheese)) $\wedge$ holds(s, quantity(x, <4, ounce>))

In order to be able to describe an object, we must also know what its STRUCTURE is. An object's structure represents the particular perspective taken upon an object, and has one of three possible values: BOUNDED INDIVIDUAL (INDIVIDUAL for short), SET, or MASS. Following from the above discussion, the structure of a physobj is not intended to correspond to any objective attribute of the physobj, but to the way in which the object is most conveniently described at some point in time: as a result, an object's structure may be different at different times. Thus, the singular noun phrase *a carrot* describes a physobj whose structure is currently *bounded individual*; the plural noun phrases *three carrots* and *three pounds of carrots* describe physobjs whose structures are both currently *set*; and the mass noun phrase *some grated carrot* describes a physobj whose structure is currently *mass*. We use the predicate *current* to indicate the state which represents the present moment in time:

(3-98) $\forall s$ current(s) $\leftrightarrow$ [time(s) = NOW]

The ingredient described as *¼ oz cheese* is then fully represented as

(3-99) $\exists x \forall s_i$ holds(s_i, substance(x, cheese)) $\wedge$
holds(s_i, quantity(x, <4, ounce>)) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, mass))

Again, for ease of manipulation, the feature structure representation is slightly different. Each feature structure effectively provides a snapshot of a given object in a particular state, thus providing a way of accessing information objects by pairs of the form <*Index, State*>. The feature structure corresponding to (3-99) is then

(3-100)
$$\left[\begin{array}{l} index = x \\ state = s_j \\ spec = \left[\begin{array}{l} substance = cheese \\ quantity = \left[\begin{array}{l} unit = ounce \\ number = 4 \end{array}\right] \\ structure = mass \end{array}\right] \end{array}\right]$$

3.3.5 Bounded Individuals and Packaging

The objects described by

(3-101) a carrot

and

(3-102) a slice of carrot

are both bounded individuals whose substance is carrot matter, but nothing in the foregoing discussion provides a way of representing the difference between the two objects. In the case of objects whose current structure is INDIVIDUAL, we must also have some notion of the way the object is *packaged.*

The packaging of an object may also change through time, and so is predicated of an object in a particular state. In a similar fashion to the examples presented above, we represent the mass-structured ingredient described simply as *sugar* as follows:

(3-103) $\exists x \forall s_i$ holds(s_i, substance(x, sugar)) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, mass))

However, if we have the ingredient described by

(3-104) a sugar cube

then we have a bounded individual, packaged in a particular way. This packaging information is relative to the current situation (since it may change), and is represented by a tuple specifying the SHAPE and the SIZE of the package:

(3-105) $\exists x \forall s_i$ holds(s_i, substance(x, sugar)) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, individual)) $\wedge$
holds(s_j, packaging(x, <cube, regular>))

As a feature structure, this is represented as

(3-106)
$$\left[\begin{array}{l} index = x \\ state = s_j \\ spec = \left[\begin{array}{l} substance = sugar \\ structure = individual \\ packaging = \left[\begin{array}{l} shape = cube \\ size = regular \end{array}\right] \end{array}\right] \end{array}\right]$$

The SIZE of an entity's packaging can be one of a number of values: for example, the ingredient described by

(3-107) one large sugar cube

is represented as

(3-108) $\exists x \forall s_i$ holds(s_i, substance(x, sugar)) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, individual)) $\wedge$
holds(s_j, packaging(x, <cube, large>))

Many ingredients which are best viewed as bounded individuals in this framework tend to be described in such a way that their packaging is not explicitly stated, as in

(3-109) a carrot

In such cases we say that the packaging used is the default for that substance, and so it is typically omitted when the object is described. By convention, the default shape of an object consisting of a particular substance is represented by the same symbol used to represent that substance. The object described by (3-109) is then represented as

(3-110) $\exists x \forall s_i$ holds(s_i, substance(x, carrot)) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, individual)) $\wedge$
holds(s_j, packaging(x, <carrot, regular>))

This can be read as saying that *a carrot* is a carrot-shaped package of carrot matter.

Note that (3-110) states basically the same information as the ISA expressions we considered at the beginning of this chapter; in general, the following equivalence holds:

(3-111) holds(s, isa(x, y)) $\equiv$ holds(s, substance(x, y))$\wedge$
holds(s, structure(x, individual))$\wedge$
holds(s, packaging(x, <y, regular>))

where y is the substance of which ys are made.

Although many ingredients come in what we have called default packagings, this is not always the case. Celery, for example, comes in either heads or sticks: we would represent *one head of celery* as

(3-112) $\exists x \forall s_i$ holds(s_i, substance(x, celery)) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, individual)) $\wedge$
holds(s_j, packaging(x, <head, regular>))

and *one stick of celery* as

(3-113) $\exists x \forall s_i$ holds(s_i, substance(x, celery)) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, individual)) $\wedge$
holds(s_j, packaging(x, <stick, regular>))

As we saw in the case of the sugar cube above, this form of representation is not restricted to 'natural' packagings: we can also represent *an onion ring* as

(3-114) $\exists x \forall s_i$ holds(s_i, substance(x, onion)) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, individual)) $\wedge$
holds(s_j, packaging(x, <ring, regular>))

We represent ingredients such as *a garlic clove* and *a fennel bulb* in the same way.

Another way in which ingredients can be described reflects the fact that certain individuals are composed of parts which are given specific names: so, for example, an avocado has as its parts a skin, a stone and a quantity of avocado flesh. We represent entities which are parts of other entities as having a PART attribute. The parts of a particular avocado are then represented by the following three KB entities:

(3-115)
$$\left[\begin{array}{l} index = x_1 \\ state = s \\ spec = \left[\begin{array}{l} structure = individual \\ packaging = \left[\begin{array}{l} shape = skin \\ size = regular \end{array}\right] \\ part = skin \\ ancestor = x \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{l} index = x_2 \\ state = s \\ spec = \left[\begin{array}{l} structure = individual \\ packaging = \left[\begin{array}{l} shape = stone \\ size = regular \end{array}\right] \\ part = stone \\ ancestor = x \end{array}\right] \end{array}\right]$$

$$\left[\begin{array}{l} index = x_3 \\ state = s \\ spec = \left[\begin{array}{l} structure = mass \\ substance = avocado \\ part = flesh \\ ancestor = x \end{array}\right] \end{array}\right]$$

Note that only the third of these entities has its SUBSTANCE specified as *avocado*: this reflects the fact that the flesh itself may be referred to as *the avocado*, but that the stone and the skin may not. We do not encode the fact that an entity has parts directly as a property of that entity: rather, we have axioms that allows us to infer the existence of those parts whenever necessary. So, for example, the following axiom allows us to infer the existence of a stone given an avocado:

(3-116) $\forall x$ substance(x, avocado) $\wedge$ structure(x, individual) $\wedge$
packaging(x, <avocado, regular>) $\supset$
$\exists x_1$ structure(x_1, individual) $\wedge$ part(x_1, stone) $\wedge$
packaging(x_1, <stone, regular>) $\wedge$ ancestor(x_1, x)

3.3.6 Sets of Objects

The third kind of object we have to be able to represent is the plural individual, or SET. A set is any physobj that is made up of a number of other physobjs, although we do

not need to be aware of the identity of these constituent physobjs.

There are a number of ways in which sets can be described. In those situations where we are not aware of the constituents' identities, as in

(3-117) 3lbs of carrots

or, more generally, where all the constituents are the same, as in

(3-118) 3 carrots

where the constituents could be listed separately but with much duplication of information, we may quantify over the elements of the set in order to specify their properties.

Sets specified by Quantity

The ingredient described as *3lbs of carrots* is represented as follows:

(3-119) $\exists x \forall s_i$ holds(s_i, quantity(x, <3, pound>)) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, set)) $\wedge$
[$\forall y$ holds(s_j, element(y, x)) $\supset$
holds(s_j, substance(y, carrot)) $\wedge$
holds(s_j, structure(y, individual)) $\wedge$
holds(s_j, packaging(y, <carrot, regular>))]

The feature structure corresponding to this is as follows:

(3-120)
$$\begin{bmatrix} index = x \\ state = s_j \\ spec = \begin{bmatrix} structure = set \\ quantity = \begin{bmatrix} unit = pound \\ number = 3 \end{bmatrix} \\ elements = \begin{bmatrix} substance = carrot \\ structure = individual \\ packaging = \begin{bmatrix} shape = carrot \\ size = regular \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

In the previous example, we specified the amount of the set using a statement of *quantity*, just as in the case of mass physobjs. Sets may also, however, be specified by means of their CARDINALITY. In such cases, note that the cardinality is state-dependent, since a quantity of some substance can be repackaged into different sets at different times. Thus, the ingredient described as

(3-121) 3 carrots

can be represented as follows:

(3-122) $\exists x \exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, set)) $\wedge$
holds(s_j, cardinality(x, 3)) $\wedge$
[$\forall y$ holds(s_j, element(y, x)) $\supset$
holds(s_j, substance(y, carrot)) $\wedge$
holds(s_j, structure(y, individual)) $\wedge$
holds(s_j, packaging(y, <carrot, regular>))]

The corresponding feature structure is

(3-123)
$$\begin{bmatrix} index = x \\ state = s_j \\ spec = \begin{bmatrix} structure = set \\ cardinality = 3 \\ elements = \begin{bmatrix} substance = carrot \\ structure = individual \\ packaging = \begin{bmatrix} shape = carrot \\ size = regular \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Just as in statements of quantity, we can state cardinality as a range. Thus, the cardinality of *3–4 carrots* is represented as follows:

(3-124) holds(s, cardinality(x, range(3, 4)))

where s is the current state.

Cardinality need not be numeric: it may also be specified as one of a limited number of *vague* values, so that we can represent the cardinality of the set described by *a few*

nuts in state s as follows:[5]

(3-125) holds(s, cardinality(x, few))

Sets specified by Constituency

Finally, the constituents of a set need not all be of the same type. In such cases, we can explicitly list the constituents of the set. Given our very general definition of a physobj, we have an equally general definition of what it is to be a constituent of a physobj:

(3-126) Given a physobj x, any portion of matter which is a proper part of the portion of matter that makes up x is also a physobj, and is said to be a CONSTITUENT of x.

The number of ways of partitioning a given physobj into different sets of constituents is thus only limited by our ability to break an object into smaller and smaller objects. In practice, however, only certain partitionings are of interest to us. As before, we use language as our guide to which partitionings are of use.

The simplest examples of this are objects which are themselves conjunctions of other objects, as in

(3-127) 350g mixed raisins, sultanas and currants

This is represented as a set that itself consists of three sets.

[5]The formalism described here only deals with existentially quantified ingredients; thus, no representation is offered for expressions like *few nuts*.

(3-128) $\exists x \forall s_i$ holds(s_i, quantity(x, <350, grams>)) $\wedge$
holds(s_i, constituents(x, [x_1, x_2, x_3])) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, set)) $\wedge$
holds(s_j, structure(x_1, set)) $\wedge$
[$\forall z$ holds(s_j, element(z, x_1)) $\supset$
holds(s_j, substance(z, raisin)) $\wedge$
holds(s_j, structure(z, individual)) $\wedge$
holds(s_j, packaging(z, <raisin, regular>))] $\wedge$
holds(s_j, structure(x_2, set)) $\wedge$
[$\forall z$ holds(s_j, element(z, x_2)) $\supset$
holds(s_j, substance(z, sultana)) $\wedge$
holds(s_j, structure(z, individual)) $\wedge$
holds(s_j, packaging(z, <sultana, regular>))] $\wedge$
holds(s_j, structure(x_3, set)) $\wedge$
[$\forall z$ holds(s_j, element(z, x_3)) $\supset$
holds(s_j, substance(z, currant)) $\wedge$
holds(s_j, structure(z, individual)) $\wedge$
holds(s_j, packaging(z, <currant, regular>))]

Notice that the constituents of the top level object are constituents not only in the current state but in all states: this reflects the fact that the constituent objects in question are physical constituents irrespective of whatever other properties are true of the objects. As a result, the top level object is represented by the feature structure in (3-129):

(3-129)
$$\begin{bmatrix} index = x \\ state = s_j \\ spec = \begin{bmatrix} structure = set \\ quantity = \begin{bmatrix} unit = gram \\ number = 350 \end{bmatrix} \\ constituents = [x_1, x_2, x_3] \end{bmatrix} \end{bmatrix}$$

with the constituent objects being represented by distinct feature structures.

Hayes distinguishes two types of objects in the world on the basis of their composition [Hayes 1985:481]. HOMOGENEOUS OBJECTS are those objects which are composed of a single piece of homogeneous stuff, and which can be broken or divided into pieces. COMPOSITE OBJECTS, on the other hand, are those objects which are made out of pieces each of which is an object, and which can be disassembled and assembled. This distinction cuts across the structural distinctions described here. In our framework, homogeneous objects will in general be viewed as having MASS or INDIVIDUAL structure, whereas composite objects will generally be viewed as having the structure SET; but not

exclusively so, since Hayes' distinction is based on 'naive ontology' rather than the way in which things are described.

3.3.7 Representing Simple Properties

Simpler properties of objects are represented in a straightforward way. Thus, an ingredient specified as

(3-130) a ripe banana

is represented as follows

(3-131) $\exists x \forall s_i$ holds(s_i, substance(x, banana)) $\wedge$
$\exists s_j$ current(s) $\wedge$ holds(s_j, structure(x, individual)) $\wedge$
holds(s_j, packaging(x, <banana, regular>) $\wedge$
holds(s_j, ripe(x, +))

The corresponding feature structure is

(3-132)
$$\left[\begin{array}{l} index = x \\ state = s_j \\ spec = \left[\begin{array}{l} substance = banana \\ packaging = \left[\begin{array}{l} shape = banana \\ size = regular \end{array}\right] \\ structure = individual \\ ripe = + \end{array}\right] \end{array}\right]$$

Attributes such as *ripe* are binary-valued, the possible values being '+' and '−'. We carry this representation over to the FOPC description of the structures, hence the use of *ripe*(x, +) instead of the simpler form *ripe*(x).

Many of the properties predicated of objects are true of those objects by virtue of some processing having been carried out. In the simplest cases, we know that some processing has been applied to an object, but we do not know anything about the object's previous state, as in

(3-133) some grated carrot

This is represented, analogously to the previous example, as:

(3-134) $\exists x \forall s_i$ holds(s_i, substance(x, carrot)) $\wedge$
$\exists s_j$ current(s_j) $\wedge$ holds(s_j, structure(x, mass))
holds(s_j, grated(x, +))

However, we often also need to represent information about the object in different states. In the case of

(3-135) one carrot, grated

the representation for the object at the time it is described is exactly the same as in the case of *some grated carrot*; note, however, that this representation does not indicate how many carrots were involved. We also need to represent the object as it was *prior* to the grating process:

(3-136) $\exists x \forall s_i$ holds(s_i, substance(x, carrot)) $\wedge$
$\exists s_j$ time(s_j) < NOW $\wedge$ holds(s_j, structure(x, individual)) $\wedge$
holds(s_j, packaging(x, <carrot, regular>))

We will return to the representation of properties resulting from processing once the representation of eventualities has been addressed in the next section of this chapter.

3.4 Eventualities

In the previous section, we presented the notation used for the representation of objects. In this section, we describe the representation of the other basic entity type encompassed by our ontology: the EVENTUALITY.

First, we relate objects and eventualities, before going on to describe the representation of eventualities. We then briefly review the principal ideas underlying the notion of a PLAN used in AI, and relate this to the notion of an eventuality presented here. We then show how the representation of eventualities permits the representation of derived objects and objects which have had processing applied to them. Finally, the relationship between hierarchical planning and the decomposition of actions is discussed.

3.4.1 Eventualities and Objects

Objects do not exist in a vacuum: they participate in relations which we may informally call states and events. In the ontology developed here, any collection of objects and the

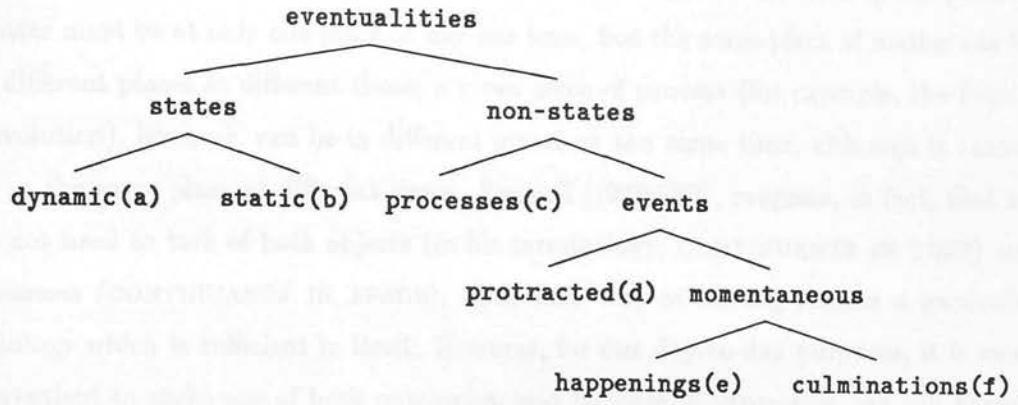


Figure 3.1: A Taxonomy of Eventualities (from Bach [1986:6])

state or event in which they participate is referred to as an EVENTUALITY, after Bach [1986]. Bach presents a more complete taxonomy of eventualities derived from Carlson [1981]: this is shown in figure 3.1. Typical examples of each kind of eventuality are as follows (the examples are Bach's own):

(3-137) a sit, stand, lie +LOC
b be drunk, be in New York, own x, love x, resemble x
c walk, push a cart, be mean (agentive)
d build x, walk to Boston
e recognize, notice, flash once
f die, reach the top

In the recipe domain, we simplify things by dealing only with Bach's general category of EVENTS (i.e., eventualities of types *d*, *e* and *f* above); however, the representation can in principle be extended to deal with the other categories. In particular, the relationship between events and the kinds of states discussed in the previous section is not fully worked out here, although if we were to adopt Bach's analysis in its entirety, we would need to take the view that (in our terms) the 'process stuff' underlying any given event can be viewed from a state perspective, and vice versa. Working out the ramifications of this approach remains as future work.

It has often been noted that strong parallels can be drawn between the logic of objects

and the logic of events. Mayo [1961] argues that events are ontologically the exact reverse of material objects with respect to time and space. Thus, a given piece of matter must be at only one place at any one time, but the same piece of matter can be in different places at different times; a given piece of process (for example, the French Revolution), however, can be in different places at the same time, although it cannot be at the same place at different times. Zemach [1979:69ff], suggests, in fact, that we do not need to talk of both objects (in his terminology, CONTINUANTS IN TIME) and processes (CONTINUANTS IN SPACE), since each way of talking reflects a particular ontology which is sufficient in itself. However, for our day-to-day purposes, it is more convenient to make use of both ontologies; and for ease of expression, we will borrow from both in the ontology presented here. Nevertheless, as we will see, it turns out that there are many similarities between our consideration of objects and events.

3.4.2 The Representation of Eventualities

Just as in the case of objects, we represent each eventuality by an index, and then predicate properties of that index as a means of representing information about the eventuality in question. We use the indices e, f, g and subscripted versions thereof for eventualities.

In addition, just as a physobj has a specification of SUBSTANCE, so too does each eventuality. The substance of an eventuality is the 'process stuff' that it consists in; thus, an instance of cycling can be represented as

(3-138) substance(x, cycling)

which is equivalent to the feature structure

(3-139)
$$\left[\begin{array}{l} index = x \\ spec = \left[\ substance = cycling\ \right] \end{array}\right]$$

Just as for the substances that make up physical objects, the substances that make up eventualities are arranged in a taxonomic hierarchy;[6] so, for example, we know that cycling is a kind of moving:

[6] See Lehrer [1969, 1972] for an analysis of the semantic structure of cooking vocabulary.

(3-140) ako(cycling, moving)

In order to describe an eventuality, we have to know how it is to be viewed; so, again just as for physobjs, we have a specification of STRUCTURE. Here our treatment is somewhat simpler than in the case of physobjs, since we deliberately restrict ourselves to one particular eventuality structure, the EVENT, and we assume that the structure of an eventuality is fixed and unchanging.

Each eventuality also has a beginning and an end: we say that it OCCURS between a begin-state and an end-state. If an eventuality begins in the state s_0 and ends in the state s_1, we write

(3-141) occurs(s_0, s_1, e)

with the proviso that

(3-142) time(s_0) $\leq$ time(s_1)

Finally, each eventuality has one or more arguments corresponding to the participants in the eventuality: there are a finite set of possible participants in an eventuality, each being identified by the ROLE it plays in the eventuality (see, for example, Fillmore [1968] and Bruce [1975]). The particular roles played by the participants in an eventuality will depend on the nature of the eventuality: thus, a sleeping event will have an agent but no object, whereas a shutting event will have both an agent and an object (both will, of course, have other participants as well). We do not need to explicitly state the identity of all the participants: as a result, an eventuality will typically only be partially specified.

As an example, we represent the eventuality describe by the sentence

(3-143) Judy shut the door.

as

(3-144) $\exists e, s_0, s_1 \forall s_i$ occurs(s_0, s_1, e) $\wedge$ holds(s_i, substance(e, shutting)) $\wedge$
holds(s_i, structure(e, event)) $\wedge$
holds(s_i, agent(e,j)) $\wedge$ holds(s_i, object(e, d))

where j and d are the indices for Judy and the door respectively: the representation of the participating individuals is omitted here. The corresponding feature structure is

then

(3-145)
$$\left[\begin{array}{l} index = e \\ occurs = \left[\begin{array}{l} begin = s_0 \\ end = s_1 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = shutting \\ structure = event \\ participants = \left[\begin{array}{l} agent = j \\ object = d \end{array}\right] \end{array}\right] \end{array}\right]$$

Again, for convenience of manipulation, the feature structure representation of the information is in a slightly more structured form than the FOPC representation.

The similarity between the representation of objects and eventualities allows the possibility of generating nominalizations. The eventuality described by the sentence

(3-146) Judy's shutting the door upset William.

is represented as

(3-147)
$$\begin{array}{l} \exists e_0, e_1, s_0, s_1, s_2, s_3 \forall s_i \text{ holds}(s_i, \text{eventuality}(e_0)) \wedge \\ \quad \text{occurs}(s_0, s_1, e_0) \wedge \text{holds}(s_i, \text{substance}(e_0, \text{shutting})) \wedge \\ \quad\quad \text{holds}(s_i, \text{structure}(e_0, \text{event})) \wedge \\ \quad\quad\quad \text{holds}(s_i, \text{agent}(e_0, j)) \wedge \text{holds}(s_i, \text{object}(e_0, d)) \wedge \\ \quad \text{occurs}(s_2, s_3, e_1) \wedge \text{holds}(s_i, \text{substance}(e_1, \text{upsetting})) \wedge \\ \quad\quad \text{holds}(s_i, \text{structure}(e_1, \text{event})) \wedge \\ \quad\quad\quad \text{holds}(s_i, \text{agent}(e_1, e_0)) \wedge \text{holds}(s_i, \text{object}(e_1, w)) \wedge \\ \quad \text{time}(s_0) \leq \text{time}(s_2) \end{array}$$

where w is the index corresponding to William, and the agent of the upsetting event is the embedded shutting event (in other words, the shutting event causes the upsetting event). Notice that the only temporal constraint specified is that the shutting event did not begin after the upsetting event. Thus, the sentence is compatible with a situation where it was Judy's turning the door handle at the very beginning of the shutting event that upset William (perhaps he was hanging on to the other side), and also with a situation where it was the bang made by the door closing that upset William.

3.4.3 Plans

We can view a recipe as a prescribed sequence of eventualities (specifically, events) that must take place for some desired goal state to be reached: in other words, it is a specification of a plan of action. The system described in the rest of this thesis generates descriptions of recipes viewed as plans of action, and so, before going on to show how this relates to our representation of objects and eventualities, it will be of value to review some of the essential elements of planning as it is conceived within AI.[7]

Planning has been an active area of research within artificial intelligence for over two decades, and planning systems technology has achieved a fair degree of sophistication as a result. The basic elements of any planning system are: some specification of an initial world state; an explicitly specified goal state; and a set of operators that can be used to achieve the transitions between world states. The task of the planning system is to specify some sequence of actions which provide a path from the initial state to the goal state.

Operators, in their simplest form, consist of a specification of some PRECONDITIONS and some POSTCONDITIONS: i.e., those facts about the world that must hold *before* the action represented by the operator can be carried out, and those facts about the world that will be true *after* the action has been carried out. By chaining operators either backwards or forwards between world states, a planning system can determine which sequence of actions is required in order to achieve the goal state. As an example, consider the simple blocks world scenario shown in figure 3.2. The initial state here might be represented as follows:

(3-148) $\text{on}(b_1, \text{table}) \wedge \text{on}(b_2, b_1) \wedge \text{clear}(b_2, +) \wedge \text{on}(b_3, \text{table}) \wedge \text{clear}(b_3, +)$

and the goal state as follows:

(3-149) $\text{on}(b_1, b_2) \wedge \text{on}(b_2, b_3) \wedge \text{on}(b_3, \text{table}) \wedge \text{clear}(b_1, +)$

Suppose our planner has two operators at its disposal, called PICKUP and PUTON. The PICKUP operator takes an object as argument and is defined as follows:

[7] For a step-by-step introduction to the basic issues in planning, see chapter 9 in Charniak and McDermott [1985]; for an overview of the planning literature, see Tate [1985].

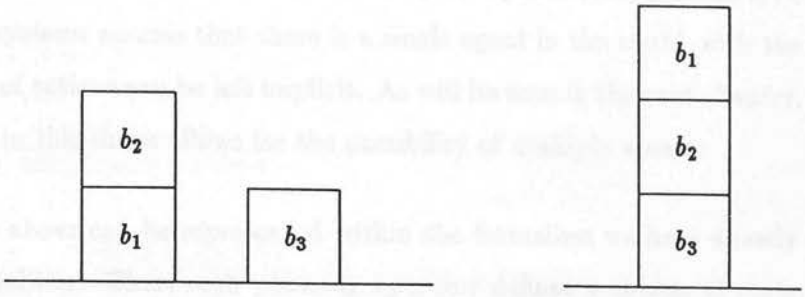


Figure 3.2: A Simple Block Stacking Problem

(3-150)

Action:	Do(A,PICKUP(x))
Preconditions:	$on(x, SomethingElse)$
	$clear(x, +)$
Postconditions:	$holding(A, x)$
	$clear(x, -)$
	$\neg on(x, SomethingElse)$

The PUTON operator takes an object x and a location y as arguments, and is defined as follows:

(3-151)

Action:	Do(A,PUTON(x, y))
Preconditions:	$holding(A, x)$
	$clear(y, +)$
Postconditions:	$clear(x, +)$
	$on(x, y)$
	$\neg holding(A, x)$
	$clear(y, -)$

Then, the goal state can be achieved by applying the following sequence of actions in the initial state:[8]

(3-152) $\text{Do}(A, \text{PICKUP}(b_2)) \wedge \text{Do}(A, \text{PUTON}(b_2, b_3)) \wedge$
$\text{Do}(A, \text{PICKUP}(b_1)) \wedge \text{Do}(A, \text{PUTON}(b_1, b_2))$

Following Cohen and Levesque [1980], we define an action term to be an eventuality from which the agent has been abstracted, thus allowing the planner to ignore the identity of

[8]Strictly speaking, the sequence of actions should be explicitly ordered, a fact not represented in the expression shown here.

the agent when this is not of importance. We notate the action term corresponding to a particular type of event by writing the name of that event in upper case: so, events which are 'putting-on' events are described by the operator whose name is PUTON. Most planning systems assume that there is a single agent in the world, with the result that the agent of actions can be left implicit. As will be seen in the next chapter, the system described in this thesis allows for the possibility of multiple agents.

All of the above can be represented within the formalism we have already introduced for eventualities. Thus, each planning operator defines a change of state, where the preconditions provide a partial description that must be true of the state in which the operator is applied, and the postconditions provide a partial description of the state that results from applying the operator. In the above example, the sequence of actions resulting in the goal state is then described as follows:

(3-153) $\exists s_0, s_1, s_2, s_3, s_4, e_0, e_1, e_2, e_3, b_1, b_2, b_3, A \forall s_i$
$\text{occurs}(s_0, s_1, e_0) \wedge \text{holds}(s_i, \text{substance}(e_0, \text{picking-up})) \wedge$
$\text{holds}(s_i, \text{agent}(e_0, A)) \wedge \text{holds}(s_i, \text{object}(e_0, b_2)) \wedge$
$\text{occurs}(s_1, s_2, e_1) \wedge \text{holds}(s_i, \text{substance}(e_1, \text{putting-on})) \wedge$
$\text{holds}(s_i, \text{agent}(e_1, A)) \wedge \text{holds}(s_i, \text{object}(e_1, b_2)) \wedge$
$\text{holds}(s_i, \text{destination}(e_1, b_3)) \wedge$
$\text{occurs}(s_2, s_3, e_2) \wedge \text{holds}(s_i, \text{substance}(e_2, \text{picking-up})) \wedge$
$\text{holds}(s_i, \text{agent}(e_2, A)) \wedge \text{holds}(s_i, \text{object}(e_2, b_1)) \wedge$
$\text{occurs}(s_3, s_4, e_3) \wedge \text{holds}(s_i, \text{substance}(e_3, \text{putting-on})) \wedge$
$\text{holds}(s_i, \text{agent}(e_3, A)) \wedge \text{holds}(s_i, \text{object}(e_3, b_1)) \wedge$
$\text{holds}(s_i, \text{destination}(e_3, b_2))$

We permit a more concise form of the above:

(3-154) $\exists s_0, s_1, s_2, s_3, s_4, b_1, b_2, b_3, A$
$\text{occurs}(s_0, s_1, \text{Do}(A, \text{PICKUP}(b_2))) \wedge$
$\text{occurs}(s_1, s_2, \text{Do}(A, \text{PUTON}(b_2, b_3))) \wedge$
$\text{occurs}(s_2, s_3, \text{Do}(A, \text{PICKUP}(b_1))) \wedge$
$\text{occurs}(s_3, s_4, \text{Do}(A, \text{PUTON}(b_1, b_2)))$

We can axiomatise the preconditions of the PICKUP operator as follows:[9]

(3-155) $\forall A, s_1, s_2, x \ \text{occurs}(s_1, s_2, \text{Do}(A, \text{PICKUP}(x))) \supset$
$\text{holds}(s_1, \text{clear}(x, +)) \wedge \exists y \ \text{holds}(s_1, \text{on}(x, y))$

Similarly, the postconditions of the PICKUP action are axiomatised as follows:

[9] This is essentially the same as the approach taken by Appelt [1986:42]

(3-156) $\forall A, s_1, s_2, x$ occurs$(s_1, s_2,$ Do$(A,$ PICKUP$(x))) \supset$
holds$(s_2,$ holding$(A, x)) \wedge$ holds$(s_2,$ clear$(x, -)) \wedge$
$\neg\exists y$ holds$(s_2,$ on$(x, y))$

3.4.4 Actions and Eventualities

Changing an Object's State

The basic ideas described above can be applied fairly straightforwardly to the domain of cooking. Some cooking operators are very simple, and do no more than change the state of an object in much the same way as the blocks world operators described above. Take, for example, the PEEL operator, which takes an ingredient and peels it. Recall from the previous section that when an ingredient is peeled, it changes state; thus, the PEEL operator is specified as follows.

(3-157)

Action:	Do(A,PEEL(x))
Preconditions:	$peeled(x, -)$
Postconditions:	$peeled(x, +)$

In reality, things are a little more complex than this. In particular, some objects only apply directly to individuals: thus, if the PEEL operator is applied to an ingredient whose structure is SET, it is the *element* specification that changes rather than that of the set itself.

Changing an Object's Structure

Some operators change the structure of an object: since the structure is a property of an object like any other, such operators also have very simple definitions. Thus, the GRATE operator is defined as follows:

(3-158)

Action:	Do(A,GRATE(x))
Preconditions:	$grated(x, -)$
Postconditions:	$grated(x, +)$
	$structure(x, mass)$

In other words, grating something changes it from an object which is perceived as a bounded individual to one which is perceived as a mass.

Deriving New Objects

Some operators are more complex, in that they may create new objects. There are two general ways in which this can be done: either by combining two or more objects, or by breaking an existing object into parts. Both kinds of operators change the object population of the world: modelling the use of these operator requires that we introduce the notion of a WORKING SET, which is a list of the identifiably distinct objects in the domain at any point in time. The working set is itself represented as a distinguished physobj, whose constituents are just the ingredients in the recipe. Thus, for any state s, we have a proposition of the following form:

(3-159) holds(s, constituents($WorkingSet$, $[x_0, x_1, \cdots, x_n]$))

where x_0 through x_n are the identifiably distinct objects in the domain in state s. Operators which create new objects manipulate the working set, adding and/or removing objects.

The simplest operator that combines two or more objects is called ADD: this takes a base object and one or more objects that are to be added to this base, resulting in a new object. The operator is defined as follows:

(3-160)

Action:	Do(A,ADD(x, y))
Preconditions:	$x \in WorkingSet$
	$y \in WorkingSet$
PostConditions:	$constituents(z, [x, y])$
	$x \notin WorkingSet$
	$y \notin WorkingSet$
	$z \in WorkingSet$

A simple example of an operator which creates a number of objects from a single object is the HALF operator, defined as follows:

(3-161)

Action:	Do(A, HALF(x))
Preconditions:	$x \in WorkingSet$
PostConditions:	$part(x_1, half)$
	$part(x_2, half)$
	$ancestor(x_1, x)$
	$ancestor(x_2, x)$
	$x \notin WorkingSet$
	$x_1 \in WorkingSet$
	$x_2 \in WorkingSet$

Thus, when a new object is added to the working set, we also have some indication of which object it was derived from. As a more complex example of this kind of operation, consider the process of obtaining the kernels from an ear of corn. We will call this operation GET-KERNELS, although it has no particular name in English. This is defined as follows:

(3-162)
Action:	Do(A, GET-KERNELS(x))
Preconditions:	$x \in$ *WorkingSet*
PostConditions:	*structure*(y, *set*)
	ancestor(y, x)
	$[\forall z : z \in y \supset$ structure(z, individual)$\wedge$
	packaging(z, <kernel, regular>)]
	$x \notin$ *WorkingSet*
	$y \in$ *WorkingSet*

Thus, the process of getting the kernels from an ear or corn removes the ear of corn in question from the working set, and adds an object which is a set of kernels. The use of operators like GET-KERNELS allows us to represent derived objects: the object described as

(3-163) the kernels from a fresh ear of corn

is then represented as

(3-164)
$$\begin{aligned}
&\exists x, s_0, s_1, e, \forall s_i \text{ holds}(s_i, \text{substance}(x, \text{corn})) \wedge \\
&\quad \text{holds}(s_0, \text{structure}(x, \text{individual})) \wedge \\
&\quad\quad \text{holds}(s_0, \text{packaging}(x, \text{<ear, regular>})) \wedge \\
&\quad\quad\quad \text{holds}(s_0, \text{fresh}(x, +)) \wedge \text{occurs}(s_0, s_1, e) \wedge \\
&\quad\quad\quad\quad \text{holds}(s_i, \text{substance}(e, \text{getting-kernels})) \wedge \\
&\quad\quad\quad\quad\quad \text{holds}(s_i, \text{object}(e, x)) \wedge \text{holds}(s_i, \text{ancestor}(y, x)) \wedge \\
&\quad\quad\quad\quad\quad\quad \text{holds}(s_i, \text{substance}(y, \text{corn})) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad \text{holds}(s_1, \text{structure}(y, \text{set})) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad [\forall z \text{ holds}(s_1, \text{element}(z, y)) \supset \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{holds}(s_1, \text{structure}(z, \text{individual})) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{holds}(s_1, \text{packaging}(z, \text{<kernel, regular>}))]
\end{aligned}$$

To gloss this in English: we have a physobj y whose structure is SET, and whose elements are all individuals which are packaged as kernels; this physobj was derived from another physobj, x, by a GettingKernels event e, where x was a quantity of fresh corn packaged as an ear.

3.4.5 Decomposition of Actions

Modern planning systems embellish the basic apparatus described above in a number of ways. Two particular embellishments are of interest here. First, in hierarchical planners,[10] we find the introduction of macro actions (also known as 'skeletal plans', 'plan schemas' and 'scripts') which aggregate together a number of more primitive actions. These higher-level actions can then be used to plan at a higher level of abstraction, with the lower level detail being filled in later. Second, we can distinguish 'linear' planners from more recent 'non-linear' planners. Linear planners adopt what is known as the 'linearity assumption', i.e., that solving the sub-goals in a task can be performed in a strictly sequential fashion. Non-linear planners, on the other hand, represent a plan as a partially-ordered network, only introducing temporal links between nodes in the plan when absolutely necessary (see Tate [1976] for a description of non-linear planning).

For simplicity, the system described in this thesis assumes that all the plans to be described are linear. Some issues relating to linearity are relevant to that part of the text generation mechanism which is responsible for the re-ordering of information prior to the generation of the final output: these are discussed in the next chapter, but are not important here. The notion of hierarchical planning is of particular importance, however.

In the examples above, all the planning operators considered were *atomic*, in the sense that they were assumed primitive and not open to decomposition. Whether or not an action is viewed as primitive depends entirely upon the context in which it is used: thus, a more detailed explanation of what it means to grate something might be required for a robot which understood only commands expressed in a language oriented towards describing simple movements of its limbs.

Within the planning literature, it is now generally accepted that constructing plans in realistic domains requires planning at different levels of abstraction. Thus, in a hierarchical planning system, we also have non-primitive operators: in addition to preconditions and postconditions, each non-primitive operator has a BODY. This provides a way of expanding the action described to produce a number of lower-level actions. Thus, we

[10] The best exposition of the basic ideas used in hierarchical planning is still that found in the original work of Sacerdoti [1974, 1977].

might have the body of the PREPARE-BEANS action defined as

(3-165) $\text{SOAK}(x) \wedge \text{DRAIN}(x) \wedge \text{RINSE}(x)$

The body of a plan operator thus provides a way of looking at the process making up the original action at a smaller grain size. More precisely, if a plan P has a body, then it can be decomposed into a number of subtasks, each of which can be represented by schemas P_1 through P_n, where

- the start state for P_1 is the same as the start state for P;
- the end state for P_n is the same as the end state for P; and
- for each i where $n - 1 > i > 0$, the end state for P_i is the same as the start state for P_{i+1}.

In other words, execution of P_1 through P_n exhausts execution of P. Recursively, each P_i may also be decomposable.

Strictly speaking, there are infinitely many ways in which an action can be decomposed into smaller events: the process stuff that makes up an action can be partitioned in infinitely many ways. Here we have another similarity between objects and events (recall our discussion in the previous section of the constituency of an object). Just as, in the case of ingredients, we saw that there were difficulties in the individuation of objects, similarly there are difficulties in the individuation of actions and events. Consider, for example, the action described as

(3-166) Peel and chop the onion and carrots.

In the appropriate circumstances, we can re-express this as any of the following:

(3-167) a Prepare the vegetables.

b Peel and chop the onions.
Peel and chop the carrots.

c Peel the onions and carrots.
Chop the onions and carrots.

d Peel the onions.
Peel the carrots.
Chop the onions.
Chop the carrots.

Do we have one event, two events or four events? Similarly, does

(3-168) Add the milk, the stock and the beans.

describe three adding actions or one? Ultimately, as in the case of physobjs, it depends on how the action is viewed. In the present system, this is determined by the level of detail used in the plan that constitutes the recipe in question: thus, we say that

(3-169) Peel and chop the onions and carrots.

describes one event decomposed into four lower level events, which have been collected together for the purposes of description.

In the real world, not all event decompositions are equal. Some are preferred over others for a number of reasons:

- some events are temporally related to others;
- similarly, some events are more naturally related because they talk about the same entities;
- finally, some events are clustered because the language provides ways of talking about them.

Given that there are certain event decompositions which are of more value than others, we can view the expansions for actions provided by the bodies of plan operators as corresponding to particular ways of decomposing one action into a number of other actions.

We can represent the eventuality corresponding to an instance of the execution of the BEAN-PREPARING action as

(3-170) $\exists e, s_0, s_3 \forall s$ holds(s, substance(e, bean-preparing)) $\wedge$
occurs(s_0, s_3, e) $\supset$
$\exists e_0, e_1, e_2, s_1, s_2$ holds(s, constituents($e, [e_0, e_1, e_2]$)) $\wedge$
occurs(s_0, s_1, e_0) $\wedge$ holds(s, substance(e_0, soaking)) $\wedge$
occurs(s_1, s_2, e_1) $\wedge$ holds(s, substance(e_1, draining)) $\wedge$
occurs(s_2, s_3, e_2) $\wedge$ holds(s, substance(e, rinsing)) $\wedge$
time(s_0) $<$ time(s_1) $<$ time(s_2) $<$ time(s_2)

We noted earlier that current planning systems allow the representation of partially

ordered plans: in the present system, we always assume that a linearisation has been adopted.

3.5 The Representation Language Summarised

This section summarises the details of the representation language presented in the preceding sections.

3.5.1 Entities in General

The domain in which the system operates consists of a finite set of entities. Each entity is represented by a distinct symbolic constant called its INDEX. An entity is either a PHYSOBJ or an EVENTUALITY. Physobjs are represented by the indices x, y, z, and subscripted versions thereof; eventualities are represented by the indices e, f, g, and subscripted versions thereof.

Every entity has, in addition to its index, a SPECIFICATION. The specification of an entity provides all the information known to the system about that entity.

Every entity also has, as part of its specification, a SUBSTANCE. In the case of physobjs, the substance is the kind of matter from which the object is made; in the case of eventualities, the substance is the kind of process stuff that makes up the eventuality. There is a finite but extensible set of substances and process stuffs represented within the system by means of symbolic constants, which are organised into a taxonomic graph structure.

As part of its specification, every entity has a STRUCTURE. This corresponds to the way in which the entity is perceived.

- If the entity is a physobj, then the structure is either INDIVIDUAL, SET or MASS.
- If the entity is an eventuality, the structure is always EVENT (in a more complete system, each of the categories identified by Bach [1986] would be permitted as a value of the structure attribute; however, in the current system, a single structure is adequate).

An entity may have any number of additional properties specified as part of its specification, where those additional properties are drawn from a finite but extensible set of properties. These properties are binary valued features with + and − being the possible values.

3.5.2 Physobjs

A physobj is defined as

(3-171) any (not necessarily contiguous) collection of contiguous regions of space occupied by matter.

If it has been derived from some other physobj in the domain, a physobj has a specification of the latter physobj as its ANCESTOR.

If a physobj has structure INDIVIDUAL, then it also has a PACKAGING as part of its specification.

A packaging is a tuple consisting of a SHAPE and a SIZE. The possible values of SHAPE and SIZE are drawn from a two finite but extensible sets.

If a physobj is a mass, it may or may not have a QUANTITY.

If the physobj is a SET, it may have either a CARDINALITY, a QUANTITY, an explicit list of CONSTITUENTS, or none of these (in which case it can only be described by a bare plural).

Cardinality is specified as a numerical value, or a RANGE of numerical values where the range consists of a LOWERLIMIT and an UPPERLIMIT.

The quantity of a physobj is specified as a tuple consisting of some UNIT of weight or other form of measure, and a number; or a range consisting of two such tuples, one for the lower limit and one for the upper limit. The possible values for the UNIT feature are drawn from a finite but extensible set.

If a physobj whose structure is SET has the feature CONSTITUENTS, the value of this feature is a list of symbolic constants which are the indices of other entities.

If a physobj whose structure is SET does not have an explicit list of constituents, then it will have an ELEMENTS feature, which provides a SPECIFICATION which is true of all the elements of the physobj.

Properties hold true of physobjs in STATES. The invariant properties (SUBSTANCE and QUANTITY) hold true of an object in all states; all other properties of objects may change.

In general, any two states are linked by the occurrence of an eventuality.

There is a privileged physobj, called the WORKING SET: this has as its constituents the set of identifiably distinct objects in the recipe at a given point in time.

3.5.3 Eventualities

By analogy with physobjs, an eventuality is defined as

(3-172) any (not necessarily contiguous) collection of contiguous regions of time occupied by process stuff.

The following are only necessarily true of eventualities whose structure is EVENT.

Every eventuality OCCURS from a BEGIN state to an END state.

An eventuality may have, as part of its specification, a finite but extensible set of PARTICIPANTS. These are collected into two substructures, corresponding to the IN participants and the OUT participants: the IN participants are those participants which exist in the BEGIN state of the eventuality, and the OUT participants are those participants which exist in the END state of the eventuality. This allows us to model eventualities which result in the destruction of existing entities or the creation of new ones. Each participant role has as its value a symbolic constant that corresponds to an entity in the domain.

Corresponding to each eventuality there is an operator which defines the preconditions and postconditions of the type of action described by the eventuality, and which may also include a possible decomposition of the eventuality into its constituents.

3.5.4 A Grammar for Knowledge Base Structures

The space of possible knowledge base structures can be defined by means of a BNF-like grammar. Some explanation of the conventions used here may be helpful.

In each rule of the form

$$A ::- B_1\ B_2\ \ldots B_1$$

the left-hand side is the name of an ATTRIBUTE which appears in a structure, and the right hand side specifies the possible VALUEs of that attribute. The symbols used to specify the right-hand sides of these rules should be understood as follows.

- Values in italics, as in

  state ::- s_i

  are atomic values. Thus, this rule states that the attribute STATE has as its value some symbol s_i. A subscripted item such as s_i is intended to indicate that any symbol of this form can be used, where i is replaced by an integer.

- Values which are in roman type, as in

  occurs ::- begin end

  are themselves attributes: thus, this rule states that the attribute OCCURS has as its value a structure that itself consists of two attributes, BEGIN and END.

- Where a number of different values are possible, the symbol '|' is used to indicate disjunction. Thus, the rule

  index ::- $x_i|e_i$

  states that the attribute INDEX may have as its value either a symbol of the form x_i or a symbol of the form e_i. In this example, the value is specified as a disjunction of atomic symbols; however, attributes can also figure in such disjunctions, as in

  quantity ::- unit number | lowerlimit upperlimit

- Where there are a potentially infinite set of values, these are represented either by a gloss in brackets, as in

prop ::- *<an attribute-value pair>*

or by one or more example values followed by ellipsis, as in

out ::- result ...

- The symbol '*' attached to an item signifies that the expansion of that item in the grammar may appear zero or more times.
- An item within square brackets indicates that the item is optional.
- An item in small capitals, as in

  spec ::- structure part ancestor PROP*

  is shorthand for the expansion of that item expressed elsewhere in the grammar; i.e., it is neither an attribute or an atomic value.

The grammar for knowledge base structures is then as follows.

(3-173)

KB	::-	index state spec|index occurs spec
index	::-	$x_i \mid e_i \mid$ *<a list of indices>*
state	::-	s_i
occurs	::-	begin end
begin	::-	s_i
end	::-	s_i
spec	::-	structure substance [quantity] PROP*|
		structure substance packaging PROP*|
		structure quantity elements PROP*|
		structure cardinality elements PROP*|
		structure quantity constituents PROP*|
		structure substance participants PROP* effects|
		structure part ancestor PROP*
structure	::-	*individual|mass|set|event*
substance	::-	*<an atomic semantic category>*
quantity	::-	unit number|lowerlimit upperlimit
unit	::-	*<an atomic semantic category>*
number	::-	$1 \ldots n$
lowerlimit	::-	unit number
upperlimit	::-	unit number
PROP	::-	*<an attribute-value pair>*
packaging	::-	shape size
shape	::-	*<an atomic semantic category>*
size	::-	*<an atomic semantic category>*
elements	::-	structure substance packaging
cardinality	::-	$1 \ldots n$
constituents	::-	*<a list of indices>*
participants	::-	[agent] in out
in	::-	[object] [base] ...
out	::-	result ...
effects	::-	add delete
add	::-	*<a list of partially specified KBs>*
delete	::-	*<a list of partially specified KBs>*

3.6 Limitations

In this section, we consider some of the major problems and limitations of the approach described in this chapter.

3.6.1 Perspectives

We suggested earlier that whether we view a given quantity of matter as a mass or a bounded individual seems to depend on the context, rather than on any intrinsic property of the quantity of matter itself. In principle, any countable object can be

put through a 'Universal Grinder' [Pelletier 1979] to be made into a mass object: given something describable by *a dog*, we can convert it into a quantity of something described as *dog*. Similarly, given a 'Universal Packager', we can convert any mass object into a countable object: thus, we can convert some quantity of something described as *coffee* into some individual described as *a coffee*.

In the representation language presented in this chapter, we only permit a given physobj to have one structure at any one time; more precisely, we only permit one PERSPECTIVE on a given physobj at any one time. However, there are situations where we might wish to permit more than one perspective on a physobj at any one time. Discussing this phenomenon, Link [1983:303-304] argues that our guide in ontological matters has to be language itself:

> if we have, for instance, two expressions a and b that refer to entities occupying the same place at the same time but have different sets of predicates applying to them, then the entities referred to are simply not the same. From this it follows that my ring and the gold making up my ring are different entities.
>
> [Link 1983:304]

Similarly, he regards the following two noun phrases as denoting the same portion of matter, but wants to say that the individuals described are distinct:

(3-174) a the cards
b the deck of cards

On Link's view, an individual lies between an noun phrase and the denotation of that noun phrase. The notion of a physobj used in this thesis corresponds to Link's notion of a portion of matter; corresponding to Link's notion of an individual, we might identify the notion of a DISCOURSE ENTITY. A discourse entity could be said to provide a PERSPECTIVE on a physobj, and is then realized by means of a noun phrase. Thus, under this view, a discourse entity consists of a pair (X, P), where X is a physobj and P is a perspective. A discourse entity then provides one way (amongst potentially many) of describing a physobj. Thus, if we view a physobj from two different perspectives, we have two discourse entities. Link's *the cards* is then, under this approach, a noun phrase which describes a physobj from a set perspective; and his *the deck of cards* is a noun phrase which describes the same physobj from a bounded individual perspective. Notice

that we can also view the same physobj as a mass individual: if I am collecting waste paper and cardboard for recycling, and I find an unwanted deck of cards, I can say

(3-175) I found some card.

Similarly, the same physobj can be described either in terms of a collection or in terms of the individuals that make up that collection: compare *a bouquet garni* and *some herbs*.

Note the difference here between *a pound of carrots* and *a deck of cards*: although the surface forms appear the same, the first describes a physobj from a set perspective (where we have a measure of the set but do not know its cardinality), whereas the latter is a statement of constitution, much closer to a noun phrase like *a stick of celery*, and is therefore being viewed as a bounded individual. This distinction is borne out by the following:

(3-176) a a stick of celery
b a celery stick

(3-177) a a deck of cards
b a card deck

(3-178) a a pound of carrots
b *a carrot pound

In practice, in the recipe domain, it is only rarely that the ability to switch between multiple perspectives on a physobj is of any real benefit; thus, in the present system, only one perspective is used at a time. However, in more complex domains, it might be of value to introduce the additional distinct level of abstraction corresponding to the notion of a discourse entity just described.

It should be noted that, in the literature, another use of the term *perspective* can be found: we may call this FUNCTIONAL perspective to distinguish it from the notion discussed above, which could more properly be called STRUCTURAL perspective. Thus, one and the same individual may be seen as *a father*, *an architect*, and *an owner of a yacht*; and different properties may be applicable depending on the functional perspective taken. This point has been discussed by Grosz [1981:100], and recognized by developers of knowledge representation languages. KRL [Bobrow and Winograd 1977],

for example, provides a way of describing different roles that an entity can perform, and KL-ONE provides the notion of a NEXUS [Brachman 1979] as a way of connecting together different views of a particular individual. Thus, I can be viewed as a programmer, a cyclist, a guitar player, and so on; we may then encode the fact that I am a lazy cyclist while preventing the inference that I am therefore a lazy programmer or a lazy guitar player. McCoy [1985] notes some of the problems associated with attempts to make use of the notion of perspectives, and suggests a modified approach. The machinery required in order to make use of structural perspectives might also be applicable to functional perspectives: the major problems lie in deciding how structural and functional perspectives are related.

3.6.2 Disjunctions

As we pointed out earlier, disjunctions are surprisingly common in recipes; however, our representation does not attempt to deal with them. In this section, we explain why the current formalism does not support disjunctions, and make some observations on the usefulness of a notion of function in the representation of recipes.

We noted earlier that ingredients in recipes can be viewed as *specifications* for participating objects. On this view, we might suggest that each of the physobjs we have described is really a specification for a more abstract entity—which, for convenience, we will call a MARKER—and that the recipe instructions operate on these markers, rather than the physical instantiations of the ingredients directly. Thus, for each marker there is a corresponding specification, but the nature of the specification is not of fundamental importance: what the operators in the plan operate upon are markers. Any effects of operators, such as changes of state, will then need to be propagated onto the specifications themselves. Thus, we would need to explicitly state which markers were realised by which physobjs.

Adding this extra level of representation suggests a way of dealing with disjunctive specifications of ingredients. We can suggest that an ingredient described as

(3-179) 1 teaspoon of salt or 1 tablespoon of soy sauce

provides a number of ALTERNATE SPECIFICATIONS for a marker. These might be

represented in the following way:

(3-180) specifications(marker, $[x, y]$) $\wedge$
substance(x, salt) $\wedge$ quantity(x, <1, teaspoon>) $\wedge$
substance(y, soy-sauce) $\wedge$ quantity(y, <1, tablespoon>)

Here, the second argument to the *specifications* clause should be interpreted as a list of mutually exclusive specifications.

However, such a representation ultimately throws up yet more problems. Suppose, for example, our disjunctive specification includes a process, as in

(3-181) 75g chopped almonds or toasted cashews

We have a problem here if we want to maintain that our operators apply only to markers and not to particular specifications, since the implied *toast* operator here applies to the cashews and not to the almonds. On the other hand, if we suggest that our operators apply, instead, to specifications, then wherever a disjunctive specification appears, the arguments to our operators start to become unwieldy. In the case of the ingredient just described, an instruction to *Add the nuts* would be represented as something like

(3-182) ADD($[x \vee y]$)

At the very least, this obscures the fact that we are talking about one functional participant in the recipe.

We begin to see that we have only just brushed the surface: it is possible that we might be able to circumvent these problems if we describe the entire recipe at the functional level, rather than the physical level, along with correspondingly abstract operations upon the participants. Such an approach would help in other areas of recipe generation too. For example, many noun phrases that appear in recipes appear to be functional descriptions: thus we see references to *the seasoning*, *the sauce*, and so on. There is also a problem in the modelling of recipes concerning the creation of new objects: in the research described in this thesis, an object is assumed to be the same object through any *state* change, but if the substance is changed—for example, if two substances are added together, or an object is broken into pieces—then we bring new objects into existence. Yet, we typically find in recipes that, if we add, for example, some salt to some cooking carrots, we still refer to the result of this addition operation as *the carrots*. In the current

system, this is dealt with by assuming that, in such cases, one constituent of the object created by an addition operation is the major constituent, and using this constituent as the basis of any references generated. However, this is really only slipping a functional approach in by the back door.

Chapter 4

The Generation of Connected Discourse

In the previous chapter, we described the formalism used to represent the kinds of objects and eventualities EPICURE has to deal with. In this chapter, we describe the processes used by EPICURE to generate connected discourse about these objects and eventualities.

First, in section 4.1, we provide an overview of the various components of the system. Subsequent sections in this chapter then describe each of the stages of processing in more detail.

In section 4.2, we examine the DISCOURSE PLANNER. This component of the system addresses the question of how EPICURE decides 'what to say'. Using plans held in the PLAN LIBRARY, and in conjunction with knowledge of the hearer's capabilities stored in the HEARER MODEL, EPICURE starts out with a top level goal, and recursively expands this to a level of detail commensurate with the hearer's assumed knowledge. The result is a structured DISCOURSE SPECIFICATION.

This discourse specification is passed to the DISCOURSE GENERATOR, described in section 4.3. The discourse generator's principal function is the construction and maintenance of the basic structure of the discourse model: the approach to discourse modelling adopted here is based on the view of discourse structure to be found in recent work by Grosz and Sidner [1985, 1986], as discussed in chapter 2. The discourse generator also dispatches the individual EVENTUALITY SPECIFICATIONS that make up the discourse

specification to the CLAUSE GENERATOR. Whereas the discourse generator is concerned with the overall construction of the discourse, the clause generator is concerned with the process of constructing individual (main) clauses within the discourse.

Before a clause can be generated, the clause generator first passes the eventuality specification to be described to the DOMAIN MODELLER. In a dynamic domain such as that used in the present work, it is important that the system's WORLD MODEL always accurately reflects the true state-of-affairs: it is the domain modeller's responsibility to ensure that this is the case, by modelling the effects of the actions to be described. This process is described in section 4.3. Once the effects of an action have been modelled, the clause generator can describe that action. It may seem rather unintuitive that, in a domain where the system is acting essentially as an instructor, the effects of a requested action are modelled *before* that action is described; however, as we will see, there are situations where knowledge of the outcome of an action is required in order to be able to describe it.

Finally, the process of generating clauses is described in section 4.4. Using the information in the world model, the discourse model, and the hearer model, in conjunction with the linguistic knowledge held in the LEXICON and GRAMMAR, this takes an eventuality specification provided by the discourse generator and produces from it a surface linguistic string that describes that eventuality.

4.1 An Overview of Epicure

The overall structure of EPICURE is as shown in figure 4.1.[1] Conceptually, the system comprises four distinct processing modules (the discourse planner, the discourse generator, the clause generator and the domain modeller), and six distinct knowledge sources (the plan library, the hearer model, the discourse model, the world model, the lexicon and the grammar). Below, we describe each of these components and the roles they play in generating a connected discourse.

[1] To re-iterate the conventions used in this diagram: data structures are represented by long flat boxes, whereas process modules are represented by the more squat boxes. An arrow pointing from box A to box B is intended to show that data flows from A to B.

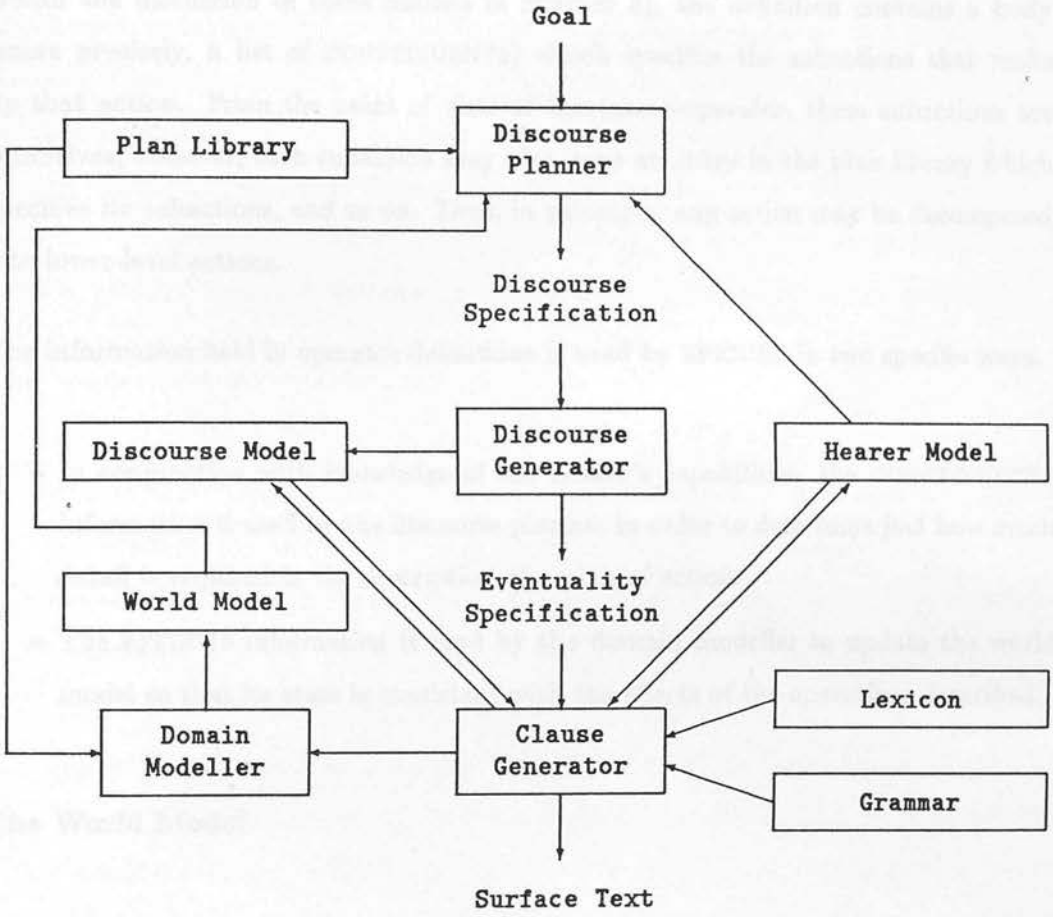


Figure 4.1: The Overall Structure of EPICURE.

The Plan Library

EPICURE starts out with a particular goal to be achieved: in the present context, this is the goal of having produced the result of a particular recipe. In order to be able to do this, EPICURE has to know what actions are necessary to achieve this goal. The actions known to EPICURE are defined, in terms of their preconditions and effects, in the PLAN LIBRARY. In addition, if the action is a macro-operator rather than a primitive action (recall the discussion of these notions in chapter 3), the definition contains a body (more precisely, a list of CONSTITUENTS) which specifies the subactions that make up that action. From the point of view of the macro-operator, these subactions are primitives; however, each subaction may also have an entry in the plan library which specifies its subactions, and so on. Thus, in principle, any action may be decomposed into lower-level actions.

The information held in operator definitions is used by EPICURE in two specific ways.

- In conjunction with knowledge of the hearer's capabilities, the CONSTITUENTS information is used by the discourse planner in order to determine just how much detail is required in the description of a plan of action.
- The EFFECTS information is used by the domain modeller to update the world model so that its state is consistent with the effects of the operations described.

The World Model

At any point in time, the domain in which the system operates is in some particular state: as we saw in chapter 3, a state can be characterised by a formula

(4-1) $P_1 \wedge P_2 \wedge \ldots \wedge P_n$

where each P_i is a proposition that is true in that state. These propositions describe properties of the entities in the domain. As the description of a plan proceeds, the properties of entities may change; some entities may even disappear, and new entities may be created. In order to describe an entity, EPICURE must know what properties are true of that entity at a given point in time. Within the system, the current state of each

entity is modelled by a knowledge base object which represents all the properties that are true of that entity in that state; thus, the world model is just that collection of such objects that describes all the entities in the domain. Whenever an action takes place, this collection of objects is updated by the domain modeller, as described in section 4.3.

The Hearer Model

The hearer model is a data structure that specifies those propositions that EPICURE assumes the hearer believes to be true. Strictly speaking, the discourse model (described below) is essentially part of the hearer model by this definition. However, for convenience, we model the two as distinct data structures. In the remainder of this thesis, we will refer to that part of the system's knowledge base which models the hearer's knowledge other than that derived directly from the discourse as the hearer model.

The hearer model contains various kinds of information:

- For each action in the domain, if the hearer is assumed to know how to carry out that action, this fact is specified in the hearer model. The discourse planner makes use of this knowledge in determining the level of detail at which a plan should be described.
- For each entity in the domain, if the hearer has any knowledge of that entity apart from that gained from the discourse, then this information is specified in the discourse model. In the current domain, this information is minimal: the hearer is assumed to know that both herself and the speaker are present in the domain (in Prince's [1981] terms, SITUATIONALLY EVOKED).
- The hearer is assumed to know the information maintained in the taxonomy of substances, so that, for example, a superordinate term such as *seasoning* can be used to describe an ingredient whose substance is *salt*.
- The hearer is also assumed to know a number of axioms of inference, so that she can infer the existence of entities that have not been explicitly introduced into the discourse. The use of this information is described in chapter 5.

Thus, the hearer model plays two distinct roles in the current system:

- it provides EPICURE with the means to determine the level of detail necessary to describe the plan; and
- it provides EPICURE with justification for using definite noun phrases to refer to a class of entities that are assumed to be mutually known outside of the discourse, either directly or by virtue of inference, and allows it to use superordinate terms when describing ingredients.

Given a goal state S, modifying the contents of the hearer model will cause EPICURE to describe the plan required to achieve that goal differently, both in terms of the level of detail at which the plan is described, and in the way in which objects participating in that plan are described.

The Discourse Planner

The discourse planner is that component of EPICURE which determines 'what to say' at the topmost level: that is, it determines which particular actions should be described. This is done using the information stored in the plan library and the hearer model, as described above; the process is described in more detail in section 4.2.

The result of this process is a tree-structured discourse specification, which corresponds to the hierarchical decomposition of the plan it describes. Some discourse-level optimization is then carried out on this structure, primarily for the purposes of making the resulting text more fluent. This involves massaging the structure to enable the use ellipsis and conjunction in the linguistic description of the plan.

The Discourse Generator

Once the optimized discourse specification has been constructed, it is passed to the discourse generator. This component has two responsibilities, which it fulfills as it walks around the discourse structure in a breadth-first fashion:

- the discourse model is constructed, as a hierarchy of focus spaces which matches the structure of the discourse specification itself; and

- the eventuality specifications which constitute the leaf nodes of the discourse specification are passed to the clause generator for realization.

This process is described further in section 4.3.

The Discourse Model

The discourse model is a repository for all the information made available in the discourse itself, and is used by the clause generator to determine how to refer to entities in the domain. In structure, the discourse model consists of a hierarchy of focus spaces: the approach to discourse modelling adopted here is based on the view of discourse structure proposed by Grosz and Sidner [1985, 1986], discussed in chapter 2. The construction of the discourse model is addressed in sections 4.3 and 4.4, and its use in referring expression generation described in chapter 5.

The Clause Generator

The process of generating clauses from eventuality specifications is described in section 4.4. The clause generator takes each basic eventuality specification provided by the discourse generator, and builds two intermediate levels of structure which result in the production of a surface linguistic string. First, a structure which represents the RECOVERABLE SEMANTIC CONTENT is constructed; from this, an ABSTRACT SYNTACTIC STRUCTURE is derived, and this is then unified with the linguistic knowledge represented in the grammar and lexicon to produce a surface string. The basic elements of this process for the overall construction of a sentence are described in the present chapter, although the details of the processes used in constructing noun phrase referring expressions are left until chapter 5.

The Domain Modeller

Before deciding on the semantic content of the description of an event, the clause generator first passes the eventuality specification to the DOMAIN MODELLER. The domain modeller retrieves the operator definition corresponding to the eventuality from the plan

library, and uses the information about the effects of the operator to update the world model appropriately. This process is described in section 4.3.

4.2 Deciding What To Say

In this section we describe the task carried out by the discourse planner. First, we review briefly the notion that language can profitably be viewed as planned behaviour. We go on discuss how the notion of hierarchical planning can be used in conjunction with a number of simple axioms in order to determine the content of a description of a plan of action. We then explain the basic mechanism used to construct a discourse specification using these axioms in conjunction with the data structures held in the plan library, and finally we describe the process of massaging this discourse specification to produce a structure more suitable for input to the discourse generation component.

4.2.1 Language as Planned Behaviour

An increasingly widespread view found in computational linguistics is that language can be viewed as planned behaviour, where the ultimate goal of that behaviour is to effect some change in the state of mind of the hearer. Of course, this view of language is not new within work in linguistics and the philosophy of language, for at its heart lies the work of Austin [1962] and Searle [1969] on speech act theory. The basic ideas of speech act theory have been formalised and examined in ever more detail by a number of researchers in computational linguistics, beginning with the early work of Bruce [1975a]; the most well known work is that of Allen, Perrault, Cohen and Levesque [Perrault and Allen 1978, Allen and Perrault 1978, Cohen and Perrault 1979, Perrault and Allen 1980, Cohen and Levesque 1980, Cohen 1981, Allen 1983]. This work has shown that speech acts can usefully be modelled as operators in a planning system, with the consequence that physical and speech acts can be integrated within the same framework.

Three major types of speech acts discussed in the literature are those of:

- informing someone of some fact;
- asking someone a question; and

- requesting someone to carry out some action.

These are typically realized by means of the declarative, interrogative and imperative sentence forms respectively. However, by means of an INDIRECT SPEECH ACT, a speaker can perform an action other than that normally associated with a particular surface form: thus, in an appropriate context, the interrogative

(4-2) Is there any salt?

may in fact convey a request to someone to pass the salt, although the surface form is more normally associated with the asking of a question.

In a planning framework, the direct speech act of informing is typically represented as something like:

(4-3)

Action:	Do(*Speaker*,INFORM(*Hearer*, *Proposition*))
Preconditions:	Believes(*Speaker*, *Proposition*)
Body:	Do(*Speaker*,S-INFORM(*Hearer*, *Proposition*))
Postconditions:	Believes(*Hearer*, Believes(*Speaker*, *Proposition*)

Similarly, the direct speech act of requesting might be represented as follows:

(4-4)

Action:	Do(*Speaker*,REQUEST(*Hearer*, *Action*))
Preconditions:	Believes(*Speaker*, CanDo(*Hearer*, *Action*))
Body:	Do(*Speaker*,S-REQUEST(*Hearer*, *Action*))
Postconditions:	Believes(*Hearer*, Believes(*Speaker*, Wants(*Action*))

Here, S-INFORM and S-REQUEST are intended to represent surface informs and requests (i.e., declaratives and interrogatives). Having these specified as the bodies of the higher level speech acts permits the representation of indirect speech acts: thus a request can be performed indirectly by means of a surface question, as we saw above. Within work in this area, the asking of a question is typically represented as a request to perform some kind of INFORM action. Thus, a yes-no question is a request to the hearer to inform the speaker whether or not some proposition is true:

(4-5) Do(Speaker,REQUEST(Hearer,INFORMIF(Speaker,Proposition)))

Speech act definitions like those given above are typically supported by subsidiary definitions, and rely on a number of simplifying assumptions. For example, in co-operative dialogue we might make the assumption that speakers do not tell lies, and that hearers

co-operate by carrying out any actions requested of them. Various researchers complicate these definitions in various ways: Cohen and Perrault [1979], for example, distinguish CANDO preconditions and WANT preconditions; Appelt [1982, 1986] distinguishes effects on knowledge (K-EFFECTS) and physical effects (P-EFFECTS). Complications of this sort are necessary in order to model the effects of these actions on the states of mind of hearers.

Within EPICURE, we are not directly concerned with the intricacies of modelling the hearer's state of mind with respect to intentions (as opposed to the hearer's state of knowledge). This permits us to make some simplifying assumptions:

The Successful Communication Assumption: Whenever the system requests an action, the hearer always understands that the speaker wants the action carried out.

The Co-operation Assumption: The hearer always co-operates: whenever the hearer is requested to carry out some action, he or she will not refuse to execute it.

The Successful Action Assumption: Actions which are requested are always carried out successfully: this simplifies the process of modelling what happens in the domain.

The System's Omniscience of the Hearer's Capabilities: EPICURE's knowledge of the hearer's capabilities is correct: if EPICURE thinks that an agent is capable of an action, then the agent is capable of that action.

It is fair to ask whether *all* language can be viewed as planned behaviour. Indeed, some work has focussed on the distinctions between planned and unplanned discourse (cf. Ochs [1979]), although the concern there is more to do with the mechanisms underlying extended discourse: the basic elements of a plan-based approach can still be relevant where smaller units of discourse are at issue. Hobbs and Evans [1979] apply the planning approach to conversational discourse and conclude that conversation can be viewed in this way, although some modifications to the planning mechanisms used in AI are necessary. Although the initial work on speech acts in a computational linguistics framework attempted to cover a very wide range of speech acts in a variety of social contexts (see Bruce [1975a]), some more recent work has focussed on TASK-ORIENTED DIALOGUES (see, for example, [Cohen 1984a]), typically in situations where an expert

describes to an assistant how to execute some task. Many of the ideas in planning seem to fit well here, particularly where the task itself has a structure that is amenable to hierarchical planning as described in chapter 3.

In the next section, we look at how the notion of hierarchical planning can play a role in deciding what to say.

4.2.2 Hierarchical Planning and Reasoning about What To Say

Sacerdoti [1977:69ff] describes how, within a dialogue, a plan may be described in increasing detail in response to the user's requests. Suppose we have a hierarchical structure representing a planned task and its subtasks: Sacerdoti's NOAH then executes the plan by means of the following algorithm.

To execute a plan of action:

- Ask the user to accomplish the action represented by the node at the top of the plan.
- If the user responds positively, assume the action has been accomplished, and so the current plan has been successfully executed.
- If the user responds negatively, assume she needs a more detailed breakdown of the execution of the action, and so execute in turn all the subtasks of the current plan.

Thus, the description of the plan unfolds recursively, to a depth determined by the user's knowledge. Of course, Sacerdoti's work was not particularly concerned with language generation, and so, in his system, asking the user to accomplish an action amounted to no more than outputting some canned information associated with the corresponding plan node. Below, we present a more formal description of a variation of Sacerdoti's basic method used by EPICURE as a means to constructing a hierarchically-structured DISCOURSE SPECIFICATION that mirrors the structure of the plan being described.

Like any planning system, EPICURE takes as input a specified initial state and a specified goal state. In the domain of cooking tasks, the specified goal state is the preparation

of food in some manner, this being the result of the execution of a recipe. In order to motivate the generation of discourses, we do not give EPICURE the capability of carrying out any actions other than making requests of someone else; as a result, the only way in which the system can achieve its goal is to describe to the hearer how to execute the cooking task in question. In order to do this, EPICURE has to build a plan that specifies the requesting actions to be carried out.

Reasoning about Action Decomposition

In order to determine what to say, EPICURE's discourse planner uses information in the hearer model and the plan library in conjunction with a number of axioms that allow the system to reason about what must be requested of the hearer.

Recall that the plan library specifies information about achieving particular goal states given particular initial states. In a proper planning system, we would use the preconditions and effects specified within the definitions of these operators in order to construct an appropriate sequence of actions to achieve the desired goal state.[2] In EPICURE, however, this aspect of the problem is not a central concern; thus, the plan library also contains very high-level operators corresponding to the particular recipes to be generated. Thus, given specific descriptions of an initial state s_0 and a goal state s_1, we stipulate that the system knows that a particular action will achieve the desired result:

(4-6) Knows(EPICURE, occurs(s_0, s_1, e))

where e might be glossed as something like *making avocado salad*, for example. This can be read as saying that the system knows that eventuality e occurring in state s_0 will result in state s_1, where states s_0 and s_1 are specified independently.

EPICURE's goal is then to ensure, or GUARANTEE, that the event e takes place. In the context of planning, it is more convenient to talk of actions rather then events. The execution of an action is an event, and so, in what follows, we will use the symbol T for the action corresponding to the event e. Intuitively, an agent A can guarantee that an

[2] In theory, at least. Cooking is very complicated, however, and even working out the preconditions and effects of the necessary primitives would be a considerable task.

action T will be executed if A is known to be able to find some way of seeing that T is executed. We say

(4-7) CanGuarantee(A,T)

This may involve the agent carrying out the action herself, or getting someone else to execute the action: the means are irrelevant. Thus, the execution of an action can be delegated to anyone who can guarantee it, who may in turn be able to delegate it to someone else, and so on.

In order to determine what must be requested of the hearer in order to ensure that the eventuality e takes place, EPICURE makes use of a number of PLANNING AXIOMS: these are axioms that control the planning of the discourse by providing a number of strategies for achieving the execution of a specified action. In order to build a plan, EPICURE must find a guarantor for each action in that plan. Thus, given a high level action T that leads to a specified goal state, the goal state can be achieved if EPICURE can guarantee performance of T:

(4-8) CanGuarantee(EPICURE,T)

The first strategy EPICURE makes use of is the SIMPLE ACTION STRATEGY. If the required action is one that an agent is physically capable of herself, then that agent can guarantee that action:

(4-9) $\forall\ A,\ T$ PhysicallyCapable(A,T) $\supset$ CanGuarantee(A,T)

We say that an agent is PHYSICALLY CAPABLE of an action if that agent knows how to perform that action, and is capable of doing so.

Information about the hearer's capabilities is one kind of knowledge recorded in the hearer model. Thus, we would represent the fact that the hearer knows how to perform SOAKING actions, and is physically capable of doing so, by the following proposition:

(4-10) PhysicallyCapable(hearer, SOAKING)

Note that this representation requires that if an agent is capable of performing an action at all, then she is capable of performing it irrespective of the object or objects to which the action is applied.

The second strategy available to EPICURE is the DELEGATION STRATEGY: if an agent B can guarantee an action, and agent A is capable of requesting B to carry out that action, then this means that agent A can guarantee the action. This is expressed by the following axiom:

(4-11) $\text{PhysicallyCapable}(A, \text{REQUEST}(B,T)) \wedge \text{CanGuarantee}(B,T) \supset$
$\text{CanGuarantee}(A,T)$

The third axiom deals with actions which can be decomposed into smaller units, and is referred to as the ACTION DECOMPOSITION STRATEGY: if an agent knows a possible decomposition for a complex action, and can guarantee each of the subactions in that decomposition, then it follows that the agent can guarantee the higher-level action.

(4-12) $\text{Knows}(A, \text{constituents}(T, [T_1, T_2, \ldots, T_n])) \wedge$
$[\forall T_i : T_i \in [T_1, T_2, \ldots, T_n] \supset \text{CanGuarantee}(A, T_i)] \supset$
$\text{CanGuarantee}(A,T)$

Some Simplifications

Together, the simple action strategy, the delegation strategy and the action decomposition strategy provide the potential for complex dialogues between multiple agents attempting to achieve a mutually held goal. To simplify things in our domain, we impose a number of restrictions. First, the *only* action that EPICURE is capable of is requesting someone else to carry out some action; so, the only statement of EPICURE's capabilities present in the knowledge base is the following:[3]

(4-13) $\forall A, T$ PhysicallyCapable(EPICURE, REQUEST(A,T))

The hearer, however, is not capable of requesting anything of the system:

(4-14) $\neg\exists T$ PhysicallyCapable(hearer, REQUEST(EPICURE,T))

This restriction prevents the construction of plans which involve infinite regresses of mutual requests (i.e., where A requests B to request A to request B to ...). If we wanted to build a system which performed multi-agent planning and permitted the possibility of dialogue, we could augment the mechanism with some means of preventing

[3] We adopt a 'negation as failure' approach to the knowledge held in the knowledge base: i.e., if a particular proposition is neither in the knowledge base nor inferrable from the contents of the knowledge base, then it is assumed to be false.

circularity, but this is beyond the scope of the present work.

4.2.3 Building a Plan

EPICURE creates an entire plan before attempting to realize any of that plan linguistically. In principle, it would be possible to interleave the two stages of plan construction and description; however, it is easier to treat them as distinct. This choice is purely an implementational issue from the standpoint of the present work.

Using the axioms described in the previous section, along with the information in the plan library and the hearer model, EPICURE can determine precisely which actions have to be described to the hearer in order to achieve a specific goal. Together, the axioms provide a way of implementing the following algorithm:

To achieve execution of a task T:

- if A is physically capable of T, then REQUEST(A, T);
- otherwise, for each $T_i \in T$, achieve execution of T_i.

As we saw in chapter 3, operators are modelled by means of knowledge base structures which correspond to partially-specified eventualities (more precisely, eventualities in which the PARTICIPANTS and BEGIN and END states have not been instantiated). If the operator is not a primitive, then the definition will also specify a decomposition by means of a CONSTITUENTS feature. So, for example, the expansion of BEAN-PREPARING as a combination of the operations of SOAKing, DRAINing and RINSing is controlled by the following structure, where the index of the top level structure is E, the BEGIN state is S_0, and the END state is S_3:[4]

[4]The INDEX and OCCURS attributes are omitted here because of lack of space.

(4-15)

$$\left[\begin{array}{l} substance = bean\text{-}preparing \\ participants = \left[\begin{array}{l} in = [object : X] \\ out = [result : X] \end{array}\right] \\ constituents = \left[\begin{array}{l} \left[\begin{array}{l} index = E_1 \\ occurs = \left[\begin{array}{l} begin = S_0 \\ end = S_1 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = soaking \\ participants = \left[\begin{array}{l} in = [object : X] \\ out = [result : X] \end{array}\right] \end{array}\right] \end{array}\right] \\ \left[\begin{array}{l} index = E_2 \\ occurs = \left[\begin{array}{l} begin = S_1 \\ end = S_2 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = draining \\ participants = \left[\begin{array}{l} in = [object : X] \\ out = [result : X] \end{array}\right] \end{array}\right] \end{array}\right] \\ \left[\begin{array}{l} index = E_3 \\ occurs = \left[\begin{array}{l} begin = S_2 \\ end = S_3 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = rinsing \\ participants = \left[\begin{array}{l} in = [object : X] \\ out = [result : X] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

If the hearer is believed to understand what it means to prepare beans, the hearer model will contain the proposition:

(4-16) PhysicallyCapable(hearer,BEAN-PREPARING)

In such a situation, the constituency information specified in the operator definition can be disregarded by the planning mechanism. However, if the hearer does not know how to prepare beans, then each constituent action becomes a goal to be achieved by EPICURE. Each of these actions will have a more complete definition, which specifies preconditions and effects as well as any further possible decomposition, as a separate entry in the plan library. When this information is required by EPICURE, it is obtained

by UNIFYING the plan structure being constructed with additional information held in the plan library. The unification of these more complete specifications with those provided in the mother goal's CONSTITUENTS slot allows EPICURE to gradually build a plan structure of increasing detail.

Note that these expansion specifications permit the distribution of participants across constituent operations, so that when an operator definition is unified with a particular working set,[5] the appropriate entities are threaded through the constituent actions. Suppose, for example, we have a recipe for making instant coffee. The input working set here might consist of three ingredients: a quantity of water, some coffee granules, and some milk; the resulting working set is a single entity, the cup of coffee. The constituent actions of this simple recipe could be described by the following discourse:

(4-17) Boil the water.
Stir in the coffee granules.
Add the milk.

The corresponding operator definition for MAKING-INSTANT-COFFEE might then look like the following, where we have elided some detail:

(4-18)
$$\left[\begin{array}{l} index = E \\ occurs = \left[\begin{array}{l} begin = S_0 \\ end = S_3 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = making\text{-}instant\text{-}coffee \\ participants = \left[\begin{array}{l} in = \left[object = [X_1, X_2, X_3] \right] \\ out = \left[result = X_5 \right] \end{array}\right] \\ constituents = [\ldots] \end{array}\right] \end{array}\right]$$

This structure specifies that a MAKING-INSTANT-COFFEE event E can be carried out beginning in a state S_0 and ending in a state S_3, applied to objects X_1, X_2 and X_3, and resulting in object X_5. The CONSTITUENTS feature specifies the subactions involved:

[5]Recall from chapter 3 that each operator is applied to a particular working set, this being the set of objects which appear within the IN slot of the PARTICIPANTS slot.

(4-19)
$$\left[\begin{array}{l} index = E_1 \\ occurs = \left[\begin{array}{l} begin = S_0 \\ end = S_1 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = boiling \\ participants = \left[\begin{array}{l} in = \left[object = X_2 \right] \\ out = \left[result = X_2 \right] \end{array}\right] \end{array}\right] \end{array}\right]$$

(4-20)
$$\left[\begin{array}{l} index = E_2 \\ occurs = \left[\begin{array}{l} begin = S_1 \\ end = S_2 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = stirring\text{-}in \\ participants = \left[\begin{array}{l} in = \left[\begin{array}{l} base = X_2 \\ addendum = X_1 \end{array}\right] \\ out = \left[result = X_4 \right] \end{array}\right] \end{array}\right] \end{array}\right]$$

(4-21)
$$\left[\begin{array}{l} index = E_3 \\ occurs = \left[\begin{array}{l} begin = S_2 \\ end = S_3 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = adding \\ participants = \left[\begin{array}{l} in = \left[\begin{array}{l} base = X_4 \\ addendum = X_3 \end{array}\right] \\ out = \left[result = X_5 \right] \end{array}\right] \end{array}\right] \end{array}\right]$$

Each of these operators may be further specified by means of distinct entries in the plan library. Note that, while the entity resulting from the BOILING action is the same entity that the action is applied to, the STIRRING-IN action results in a new entity, X_4, which then serves as one of the inputs to the ADDING action. The use of shared variables in the planning operator definitions thus ensures that, when actions are decomposed, the constituent actions will be applied to the appropriate entities.

The discourse planning algorithm terminates when the entire plan is composed of actions of which the hearer is believed to be physically capable. The result is a tree-structured plan, where the terminal nodes in the tree are those actions that must be requested of the hearer. Each such action can therefore be embedded with a REQUEST operation whose agent is EPICURE itself: the resulting DISCOURSE SPECIFICATION is thus a structured set of speech acts of the form

(4-22) $[T_1, T_2, \ldots, T_n]$

where each T_i is either a primitive speech act, or an embedded discourse specification. So, we might have a structure like the following:

(4-23) $[T_1, T_2, [T_{31}, T_{32}, \ldots, T_{3n}], T_n]$

where the structure of the discourse specification matches that of the task itself. Each primitive T_i is a REQUESTING eventuality, with the following form:

(4-24)
$$\left[\begin{array}{l} index = e \\ occurs = \left[\begin{array}{l} begin = s_0 \\ end = s_1 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = requesting \\ participants = \left[\begin{array}{l} agent = \text{EPICURE} \\ object = \left[\begin{array}{l} index = e_1 \\ occurs = [\ldots] \\ spec = [\ldots] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

where the embedded OBJECT is the requested eventuality. The representation of REQUESTs is somewhat simpler than that of physical actions, since the simplifying assumptions we made earlier remove the need to reason about the effects of speech acts on the hearer's state of mind.

To simplify the process of modelling the effects of actions in the domain, we also make the BEGIN and END states of the REQUESTING eventuality the same as the BEGIN and END states of the requested eventuality. Of course, this is hardly an accurate representation of what happens in the real world: in general, execution of a requested action begins

after the request itself has ended. However, this is of no consequence in the current framework.

4.2.4 Massaging the Discourse Specification

It has often been noted that knowledge representations used by programs which are not concerned, in the first instance, with language generation, are often far from ideal from the needs of a language generator. Thus, for example, Mellish [1988:137–138] describes a MESSAGE OPTIMISATION phase in the language generation process: the purpose of this phase is to modify and simplify the output data produced by a planner so that it is more convenient for the purposes of the generator. Mellish's optimisation rules are, in the main, not linguistically-oriented; rather, they perform a role analogous to that of CRITICS (cf. Sacerdoti [1977]) in non-linear planning systems, watching out for and removing redundancies in the plan structure itself. EPICURE, on the other hand, is primarily oriented towards discourse generation, and so any massaging of the message that it carries out always has linguistic goals in mind. Currently, this aspect of the system is very primitive, and does no more than modify the structure produced by the discourse planner to allow the clause generation process to make use of conjoined verbs.[6] Further optimisations could be added, however. In particular, if integrated with a non-linear planning system, one of the functions of the massaging process would be to determine the best order in which to present information: note that, in the context of recipe description, the order of presentation of actions does not always match the order of execution. Often, the most useful temporal ordering of the actions is sabotaged by the author of the recipe, perhaps in the interests of editorial compactness, so that actions relating to the same objects are often described together, even if they are temporally quite distant from each other. Consider the following recipe, for example:

(4-25) Cook the rice; cool.
Sprinkle apple and banana with lemon juice and add to rice.
Steep raisins in a little boiling water for half an hour to plump; drain, and add to rice with sunflower seeds, mixing well.
Fry onion in oil with curry powder for 10 minutes.

Here, the instruction to *steep the raisins* comes after the instruction to *add the apple and banana to the rice*, although it would obviously be more sensible to start steeping

[6] There are similarities here with the collapsing together of moves to be described as carried out by Davey's program, discussed in chapter 2.

the raisins at an earlier point in time.

In EPICURE, message optimization rules operate only on sequences of primitive REQUESTs. This is necessary in the current framework, since otherwise the structure of the resulting discourse specification may deviate substantially from the structure of the underlying task. Thus, we have an optimisation rule which checks to see if a sequence of eventualities has the same participants: if so, the description of these eventualities can be collected together, resulting in the generation of a number of conjoined verbs. In the case of bean preparation discussed earlier, for example, the following are all primitive actions:

(4-26) a Do(EPICURE, REQUEST(hearer, SOAK(x)))
b Do(EPICURE, REQUEST(hearer, DRAIN(x)))
c Do(EPICURE, REQUEST(hearer, RINSE(x)))

However, since all three REQUESTs are sisters in the discourse specification, they can be collected together. The result is a single REQUEST eventuality which has the following object:

(4-27)
$$\left[\begin{array}{l} index = [e_1, e_2, e_3] \\ occurs = \left[\begin{array}{l} begin = s_0 \\ end = s_3 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = [soaking, draining, rinsing] \\ participants = \left[\begin{array}{l} agent = hearer \\ in = \left[\begin{array}{l} object = x \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

As a result, instead of generating the sequence of sentences:

(4-28) Soak the butterbeans.
Drain them.
Rinse them.

we can generate the single sentence

(4-29) Soak, drain and rinse the butterbeans.

Optimization rules closer in spirit to those used by Mellish would be appropriate for collecting together clauses into complex sentences. Brée and Smit [1986], for example, specify rules for incorporating subordinate clauses into sentences: these would be par-

ticularly useful if we were to generate recipes that included more detailed descriptions of actions that incorporate information such as specifications of duration and manner. Incorporating the necessary information is far from trivial, however, and beyond the scope of the work presented here.[7] In what follows, we will only generate single clause sentences.

A final modification made to the discourse specification involves the addition of a DESCRIBE speech act to the front of the structure for each entity in the working set of the top level action in the plan. Thus, a series of structures of the following kind are added:

$$(4\text{-}30)\quad \left[\begin{array}{l} index = e \\ occurs = \left[\begin{array}{l} begin = s_0 \\ end = s_1 \end{array}\right] \\ spec = \left[\begin{array}{l} substance = describing \\ participants = \left[\begin{array}{l} agent = \text{EPICURE} \\ object = x_i \end{array}\right] \end{array}\right] \end{array}\right]$$

The complete discourse specification then has the form

$$(4\text{-}31)\quad [D_1, D_2, \ldots, D_n, R]$$

where each D_i is a DESCRIBE action, and R is either a primitive REQUEST (in the limiting case where a single action is to be requested), or a discourse specification. Thus, in principle, the working set required by each individual action could be described immediately prior to that action; however, it is more convenient to describe all the ingredients of a recipe at the beginning of a recipe; and so, in general, no DESCRIBE acts will appear other than at the top level of the discourse specification.[8]

4.3 Discourse Generation and Domain Modelling

In the previous section, we saw how EPICURE constructs a discourse specification. In this section, we show how the discourse generator uses this discourse specification to

[7] In this connection, see Karlin [1988] on the variety of semantic case roles that are common in recipes.

[8] This is analogous to a situation in programming, where we might decide to declare all variables as global, rather than local to the procedures which use them.

control the construction of a discourse model, and we describe the process whereby the effects of the actions described in the discourse specification are modelled by the domain modeller.

4.3.1 The Discourse Model

In chapter 2, we discussed the notions of salience and information status, and distinguished various degrees of assumed familiarity that might be predicated of an entity. The main point that emerged from that discussion was that the salience or information status of an entity is a crucial factor in determining the kind of linguistic expression that can be used to refer to that entity.

In this section, our primary concern is with modelling the information status of those entities which have been explicitly introduced into the discourse; i.e, those entities which have been, in Prince's [1981] terminology, TEXTUALLY EVOKED. Most computational models of discourse understanding or generation model this aspect of information status by means of a discourse model of some kind. Typically, this structure is used to determine

- when a pronoun can be used to refer to an entity; and
- when a definite referring expression, as opposed to an indefinite referring expression, can be used to refer to an entity.

In most cases, the discourse model is no more than a list of the entities which have been mentioned in the discourse. Assuming a very simple pronominalization heuristic, then, if we have a discourse model D and an intended referent x, we might have a simple algorithm along the following lines:

(4-32) if x was mentioned in the previous sentence then use a pronoun
else if $x \in D$ then use a definite noun phrase
else use an indefinite noun phrase

However, as we noted in chapter 2, another aspect of the meaning of the definite determiner is the uniqueness of the description contained in the noun phrase; the above algorithm has nothing to say about this.[9] Ideally, our discourse model should have

[9] We could also take issue with the simple approach to pronominalization incorporated here: we will

something to say about how the relevant intended context of interpretation is determined.

Below, we first note some of the requirements we would like to see met by a discourse model, and then go on to describe the structure of the discourse model used by EPICURE.

The Requirements to be met by the Discourse Model

We noted in chapter 2 that some anaphoric phenomena, such as *one*-anaphora, verb phrase ellipsis, and most instances of pronominalization, appear to obey a kind of locality constraint, in that the antecedent material required for their resolution tends to occur within the previous few sentences of the discourse. Full definite noun phrase reference, on the other hand, does not appear to be restricted in this way.

Apart from simply recording which entities have been mentioned in the discourse, we would like our discourse model to have something to say about the fact that anaphoric phenomena appear to fall into these two broad categories. One suitable distinction we might embody within the discourse model, then, is that suggested by Grosz [1977], between local and global focus. As we saw earlier, the same basic distinction is supported by the psychological evidence. Adopting this distinction then raises a number of questions:

- What part of the preceding discourse corresponds to local focus?
- What kinds of information are stored in the element of the discourse model corresponding to the local focus?
- What internal structure, if any, does the component of the discourse model corresponding to global focus have?
- What kinds of information are stored in the element of the discourse model corresponding to the global focus?

The answers we give to these questions should take account of as many of the anaphoric phenomena we saw in chapter 2 as possible. The extent to which our chosen model

return to this in chapter 5.

achieves this goal, and how it does so, will be examined in chapter 5. Below, we describe the basic structure of the model we adopt.

The Structure of the Discourse Model

Our discourse model distinguishes two principal components, corresponding to Grosz's [1977] distinction between local focus and global focus.

We will call that part of the discourse model corresponding to the local focus CACHE MEMORY. In this thesis, we assume that cache memory contains the lexical, syntactic and semantic detail of the current (major) clause being generated, and the same detail for the previous (major) clause. The significance of the clause as the primary unit of discourse modelling can be questioned. It has often been suggested that the basic unit of communication is the sentence (e.g., Winograd [1983:469]), but there are various other measures we might have chosen: apart from the contents of a single sentence, clause, or utterance, we might have specified a fixed number of words, morphemes or even phonemes. Ultimately, all of these suggestions have something to offer, but how the differing demands they make might be reconciled will not be pursued here.

As stated at the outset of this work, psychological plausibility is not one of the our aims. This does not mean, however, that work in psychology and psycholinguistics is of no relevance. We might note, then, that a considerable amount of research in this area has argued that the sentence, or clause, or some closely related unit, constitutes the basic planning unit in language production. In this connection, see the work of Bernardo [1977] on the cognitive relevance of the sentence, and Taylor [1969], Butterworth [1975], MacNeilage [1973], Danks [1977], and Lindsey [1975, 1976] on the size of planning units.

Corresponding to global focus, our discourse model will consist of a number of FOCUS SPACES. These focus spaces record the semantic content, but not the syntactic or lexical detail, of the remainder of the preceding discourse. As suggested by Grosz and Sidner's work [Grosz 1977, Grosz and Sidner 1986], some of these focus spaces will be arranged in a focus stack that models the focus of attention in the discourse; however, our discourse model must provide a history of the relevant detail from all the discourse, not just those parts currently in focus, and so we also require a structure that consists of those focus

spaces that have been removed from the focus stack.

A major motivation for the adoption of a discourse model of this kind is that it fits well with the approach taken in this thesis to the generation of discourses from underlying plans, i.e., by means of recursive expansion of plan operators into their components. As we will see, there are a number of ways in which this basic model can be used; and, as will be discussed in chapter 5, it may help to provide an explanation for long distance pronominalization.

More precisely, our discourse model has the following structure.

Cache memory consists of three parts:[10]

Current Clause Work Space: This contains the recoverable semantic structure corresponding to the current clause, and both the abstract and surface syntactic structures generated from it (the format of these structures will be described later in the present chapter).

Previous Clause Contents: This contains the recoverable semantic structure of the previous clause generated.

The Current Centre: This contains the index of the current discourse centre, intuitively similar to the notion of centering suggested by Grosz, Joshi and Weinstein [1983]: i.e., something like the focus of attention. In recipes, we take the centre to be the result of the previous operation performed: because of this, the centre may not always be realized linguistically. The effects of this will be seen later.

The global focusing component of the discourse model also consists of two distinct parts:

The Focus Stack: This contains a focus space for each discourse segment which has not been closed: each focus space consists of a set of structures, one for each entity (object or event) mentioned in that focus space. These structures are partially specified knowledge base structures, as defined in chapter 3: thus, they

[10] Our use of the term 'cache memory' is slightly different to that proposed by Guindon [1983:218], who uses the term to refer to a memory structure which contains 'a subset of the meaning units expressed in the previous sentences'. For her, cache memory is one part of short term memory: the other part, called the BUFFER, holds a representation of the incoming sentence. Thus, Guindon's short term memory is our cache memory; her cache is our previous clause contents structure; and her buffer is our current clause workspace.

are intended to represent the hearer's knowledge of the entities introduced in the discourse.

The Closed Focus Space List: This structure contains the set of focus spaces which have been popped from the focus stack as a result of their discourse segments having been closed.

Within EPICURE, the discourse model is assumed to be identical for both the speaker and the hearer. This permits us to make use of one data structure for modelling the discourse, instead of having to maintain one discourse model for the speaker, and one for the hearer. It also means that we adopt what has been called the STANDARD NAME ASSUMPTION (cf. Appelt and Kronfeld [1987:641]): that is, that the internal names used to represent real-world entities are the same for both speaker and hearer. So, if the system constructs the referring expression

(4-33) the black olives

in order to refer to the entity x, then the act of reference succeeds if the hearer associates the entity x with this description. Appelt and Kronfeld point out that, in general, this assumption is too strong, and attempt to eliminate it; however, we adopt it for the present purposes. Thus, by consulting the discourse model to ascertain the relative salience of an entity to be referred to, the speaker can assume that the entity will have a similar degree of salience in the hearer's discourse model.

Note that the assumption means that there is no scope in the present system for modelling COMMUNICATION FAILURE, whereby the discourse participants believe themselves to be talking about the same entity when in fact they are talking about different entities (in this connection, see the work by Goodman [1985, 1986] on reference repair). These concerns are beyond the scope of the present work.

4.3.2 The Discourse Generation Algorithm

Once the discourse planner has constructed the discourse specification, each element of this structure has to be passed to the clause generator. In conjunction with this, the basic elements of the discourse model must also be manipulated. Co-ordinating the construction of the discourse model with the dispatch of eventuality specifications to

the clause generator is the task of the discourse generator component.

Recall from section 4.2 that a discourse specification has the following structure:

(4-34) $DS = [D_1, D_2, \ldots, D_n, R]$

where each D_i is a DESCRIBE action, and R is either a primitive REQUEST (in the limiting case where a single action is to be requested), or a discourse specification. Given a discourse specification DS produced by the discourse planner, the process of generating a discourse proceeds as follows.

- Push a focus space for this discourse onto the focus space stack, and make it the current focus space (CFS).
- For each DESCRIBE action in DS:
  - Build a recoverable semantic structure which describes x_i, the object of the DESCRIBE action: this process is described in chapter 5. The recoverable semantic structure represents all the information that the hearer should be able to recover from the resulting noun phrase.
  - Add a knowledge base structure for x_i to CFS, where that knowledge base structure contains the information chosen for inclusion in the deep semantic structure.
  - Pass the recoverable semantic structure to the surface generator. This will construct an abstract syntactic structure which is suitable for input to the grammar: again, this process is described in chapter 5.

Once all the initial descriptions have been dealt with, we turn to the remainder of the discourse specification.

To describe R:

- If R is a primitive REQUEST, pass it to the clause generator (described later in this chapter).
- If R is a list of specifications, then do the following.
  - Push a new focus space onto the stack and make it the CFS.

- Describe each $R_i \in R$.
- Pop the focus stack and add the popped focus space to the closed focus space list.

Thus, the top level mechanism recursively builds a focus space structure that follows the embedding of the discourse structure.

4.3.3 Modelling the Domain

In order to be able to compute appropriate referring expressions, the system's world model must be accurate at the time those descriptions are computed. Ensuring that this is the case is the task of the DOMAIN MODELLER. Before the clause generator attempts to construct a clause describing an eventuality specification, it first passes the eventuality specification to the domain modeller. This models the effect of the action about to be described. This means that, when clause generation proceeds, EPICURE has at its disposal information describing both the state of the world *before* the action to be described took place, and also *after* the action to be described took place. This is necessary in the construction of clauses like

(4-35) Remove the stalks from the strawberries.

where the entity referred to by *the stalks* is not present explicitly in the domain until after the DESTALKING action has taken place.

Recall that the state of all the entities in the domain can be characterised as a conjunction of propositions of the form:

(4-36) $holds(s, P)$

where P is a predication of some property of an entity, and s is the state in which this predication holds true. A serious problem for any system which models a dynamically changing domain is the FRAME PROBLEM: that is, for any change that takes place as the result of some event, how do we model the fact that most other things do not change?

In the current framework, this problem is not too severe, since we are dealing with a

small number of entities at any one time. We thus have a FRAME AXIOM which states that, other things being equal, nothing changes: given an event e such that

(4-37) $\text{occurs}(s_0, s_1, e)$

then in the general case:

(4-38) $\forall P \ \text{holds}(s_0, P) \supset \text{holds}(s_1, P)$

In other words, everything that was true before the event took place is also true after the event took place. Of course, some things *do* change as the result of events having occurred. Within a procedural model, this poses no significant problems. EPICURE models the effects of an action by means of ADD and DELETE lists (this method dates back to the STRIPS system [Fikes and Nilsson 1971]): the ADD list specifies those propositions which have to be added to the new state description, and the DELETE list specifies those propositions which have to be removed from the state description. Thus, for the operator SOAK, we have the following information:

(4-39) Add: $soaked(x, +)$
Delete: $soaked(x, -)$

Let A be the add list for a given operation, and D be the delete list for that operation: the domain model is then updated as follows whenever an eventuality e causes a transition from state s_0 to state s_1.

- for each P such that $holds(s_0, P)$, add the information that $holds(s_1, P)$;
- for each $P \in D$, remove $holds(s_1, P)$;
- for each $P \in A$, add $holds(s_1, P)$.

Within EPICURE, the effects of an action are modelled as DELETE and ADD lists included as part of the SPEC of the operator in question. So, for example, the delete list of the SOAKING operator is as follows:

(4-40)
$$\left[delete = \left[\left[\begin{array}{l} index = X \\ state = S_2 \\ spec = \left[soaked = - \right] \end{array} \right] \right] \right]$$

Similarly, we have an ADD structure:

$$(4\text{-}41)\quad \left[add = \left[\left[\begin{array}{l} index = X \\ state = S_2 \\ spec = \left[soaked = + \right] \end{array} \right] \right] \right]$$

Thus, using the frame axioms described above, updating the world model involves creating new knowledge base structures for each object in the domain (i.e., each object in the global working set), identical to the old knowledge base structures except that the value of the STATE feature is changed from the BEGIN state of the event in question (say S_1) to the END state of that event (say S_2). Then we apply the following algorithm:

- For each element d in the event's DELETE list:
  - Obtain the knowledge base structure indexed by $<d\ index>$ and $<d\ state>$.
  - For each element in the SPEC feature in d, remove that element from the knowledge base structure (if it is there: it may not be present).
- For each element a in the ADD list:
  - Obtain the knowledge base structure indexed by $<an\ index>$ and $<a\ state>$.
  - For each element in the SPEC feature in a, add that element to the knowledge base structure.

Execution of an action also causes the current working set to be updated. The constituency of the working set is determined by the operator definition itself. Recall the MAKING-INSTANT-COFFEE operator we saw earlier:

$$(4\text{-}42)\quad \left[\begin{array}{l} index = E \\ occurs = \left[\begin{array}{l} begin = S_0 \\ end = S_3 \end{array} \right] \\ spec = \left[\begin{array}{l} substance = making\text{-}instant\text{-}coffee \\ participants = \left[\begin{array}{l} in = \left[object = [X_1, X_2, X_3] \right] \\ out = \left[result = X_5 \right] \end{array} \right] \end{array} \right] \end{array} \right]$$

All those distinct objects which appear in the IN structure are constituents of the working set at the BEGIN state, whereas all those distinct objects which appear in the OUT structure are constituents of the working set at the END state. Suppose we have a MAKING-INSTANT-COFFEE event occurring from state s_0 to state s_1; then we have

(4-43) holds(s_0, constituents(x, [x_1, x_2, x_3]))

before the event, and

(4-44) holds(s_1, constituents(x, [x_5]))

afterwards, where x is the global working set.

4.4 Generating Clauses

In this section we describe how an eventuality specification, as produced by the discourse generator, is realized as a sentence by the clause generator. The main issue here is the determination of the semantic content of the sentence to be output, along with the specification of additional information that determines ordering or particular grammatical choices. Within EPICURE, there are two intermediate levels of representation corresponding to a given surface string: we call these the RECOVERABLE SEMANTIC STRUCTURE and the ABSTRACT SYNTACTIC STRUCTURE. The distinctions between these levels of representation are described below.[11] We then go on to show how these two levels of representation are constructed, and how a unification grammar is used to realise a surface string from the abstract syntactic structure. Finally, the clause generation algorithm is summarised.

4.4.1 Levels of Representation

The input provided to the clause generator by the discourse generator is a particular kind of knowledge base (KB) structure, namely an eventuality specification which specifies a speech act. In the current context, this is always a REQUEST. The input structure is

[11] At an earlier stage in the development of this work, the recoverable semantic structure was referred to as DEEP semantic structure, and the abstract syntactic structure was referred to as SURFACE SEMANTIC structure: thus, the two intermediate levels of structure were both viewed primarily as semantic structures. However, although the abstract syntactic structure does indeed encode what we might call semantic information, it is primarily motivated by syntactic concerns, hence its name.

therefore something like the following.

$$
(4\text{-}45)\quad
\left[\begin{array}{l}
index = e \\
occurs = \left[\begin{array}{l} begin = s_0 \\ end = s_1 \end{array}\right] \\
spec = \left[\begin{array}{l} substance = requesting \\ participants = \left[\begin{array}{l} agent = system \\ object = [\ldots] \end{array}\right] \end{array}\right]
\end{array}\right]
$$

where the embedded OBJECT is the particular eventuality to be requested. Realizing this request as a surface linguistic form involves the construction of two intermediate levels of representation, as follows.

Recoverable Semantic Structure

As has often been noted, a representational formalism that is useful for one task may be inappropriate for other tasks (see, for example, Swartout [1983]). So, just as we saw in our discussion of McDonald's MUMBLE in chapter 2, in a situation where a generation system is acting as a front-end to some other application, it may be necessary to provide some mapping from the particular structures used by the application to structures that are more appropriate to the needs of the generation system itself.

In the case of EPICURE, the particular structures which are useful for modelling the domain for the purposes of planning are not the most appropriate for the generation of text. More particularly, an eventuality specification as provided by the discourse generator, since it is consists of an instantiated plan operator, may include information that is not required in the description of that eventuality: in this sense, the structure is over-specified. In another respect, the structure may be under-specified: in particular, the participants in the eventuality are specified by means of their indices alone, with no associated information attached.

Thus, given an eventuality specification, the clause generator uses a set of MAPPING RULES (described below) to build a RECOVERABLE SEMANTIC STRUCTURE. This specifies, in a relatively flat form, the semantic content of the utterance to be generated. In this structure, the indices of objects are replaced by structures which describe the

semantic content of the noun phrases that will describe these objects; and any participants and other information in the eventuality which are not to be described will have been omitted. Thus, the mechanism used to build the recoverable semantic structure represents the bridge between knowledge representation and semantic representation, where the latter is a linguistically-motivated construct. The semantic representation is intended to be domain-independent: the particular mechanism used by EPICURE to construct the recoverable semantic structure in the current domain could be replaced by a different mechanism to permit it to generate text for other applications.[12]

As the name suggests, the recoverable semantic structure corresponding to an event to be described represents the semantic content which should be recoverable from the utterance by the hearer: this does not mean that all of this semantic content need be explicitly *realized* by the clause generator. So, for example, the use of of *one*-anaphora and verb phrase ellipsis is not represented at the level of recoverable semantic structure; for related reasons, it is the recoverable semantic structure that is integrated into the discourse model.

Abstract Syntactic Structure

A second set of mapping rules is then applied to the recoverable semantic structure, resulting in the construction of an ABSTRACT SYNTACTIC STRUCTURE. This structure is much closer to the syntax of the linguistic form that will eventually be produced: it incorporates ordering information, and takes account of the scope for *one*-anaphora and other forms of ellipsis. The mapping from recoverable semantic structure to abstract syntactic structure is one-to-many: so, for example, the recoverable semantic structure will specify which illocutionary act is to be performed by the utterance being generated, but will not specify which particular surface form is to be used to realise this act.

Making use of two intermediate levels of representation in this way is not without precedent. Webber [1979] does this, for example, and for reasons not unrelated to those that motivate the distinction here: in particular, she found it useful to distinguish a level of representation which does not include anaphoric elements from one that does.

[12] As things are set up at the moment, any such application would have to provide knowledge base structures that are expressible as single clause sentences. In practice, it is probably more desirable to interface application and generation system at a higher, or more abstract, level: see, for example, the work of Moore and Mann [1979, 1982].

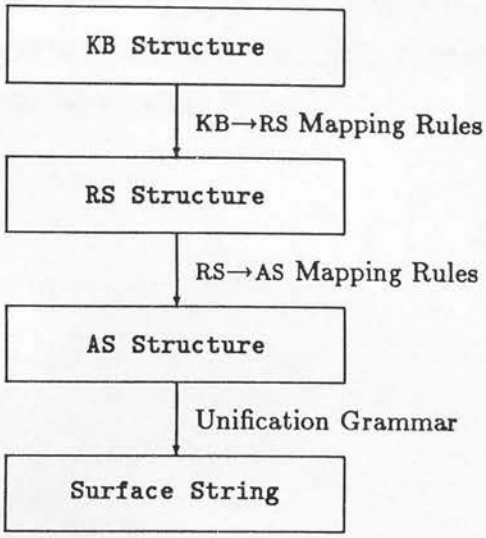


Figure 4.2: The Levels of Representation in Clause Generation

The abstract syntactic structure is then unified with the grammar, as described later in this section, resulting in the production of a surface string.

The clause generation process therefore consists of a series of transducers, as shown in figure 4.2. These levels could be compiled together, but it is conceptually simpler to deal with them as distinct processes. The actual processes involved will be described in more detail in the remainder of this section.

4.4.2 Building the Recoverable Semantic Structure

First, we describe the basic form of recoverable semantic structures; then we present some of the mapping rules that are used to construct recoverable semantic structures from knowledge base structures.

Recoverable Semantic Structure

The domain of the KB→RS mapping rules is the universe of knowledge base structures, and the range is the universe of recoverable semantic (RS) structures. The set of possible RS structures is defined by the following grammar:

(4-46)

RS	::-	index sem
index	::-	$x_i \mid e_i \mid$ *<a list of indices>*
sem	::-	illocforce tprop
illocforce	::-	*requesting, informing, ...*
tprop	::-	tense prop
tense	::-	*present, past, inf ...*
prop	::-	substance participants
substance	::-	*<a basic substance>*
participants	::-	[agent] [in] [out]
in	::-	[object] ...
out	::-	[object] ...

The recoverable semantic structures corresponding to the individual participants are not included in this grammar, but will be described in chapter 5.

So, the basic form of a recoverable semantic structure corresponding to a speech act eventuality is the following:

(4-47)

$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} illocforce = \ldots \\ tprop = \begin{bmatrix} tense = \ldots \\ prop = \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Mapping Rules

The mapping from knowledge base structure to recoverable semantic structure is one-to-many: the end result of this stage depends on the particular mapping rules used.

Mapping rules within EPICURE always take the same basic form: they specify an input pattern, and a resulting output pattern. The left hand side of each rule is some partially-specified structure in the domain of the rule; the right hand side of each rule is some partially-specified structure in the range of the rule. Instantiating a rule against a particular input structure and a particular output structure results in the output

structure becoming more fully specified as a result of the rule having applied.

Mapping rules are collected together into ordered sets, such that for any given input structure, only the first rule in each set which matches the input structure will be used. By forcing the system to backtrack, subsequent rules which match the input structure can be used, thus producing multiple output structures for a given input structure. The algorithm for building a recoverable semantic structure from a knowledge base structure then applies a number of sets of rules to the input structure in turn.

Within the current context, there are effectively four sets of rules that are used, with the following functions:

1. Build the basic recoverable semantic structure for the utterance to be generated, specifying the illocutionary force to be used.
2. Determine the tense to be used in the utterance.
3. Determine which participants in the eventuality are to be mentioned or described in the utterance.
4. Build recoverable semantic structures corresponding to the participants to be described.

This last step is essentially recursive, and will be described in detail in chapter 5. The first three steps are outlined below.

Determining the Illocutionary Force

The illocutionary force to be used depends straightforwardly upon the substance of the eventuality specification itself (recall that the eventuality specification passed to the clause generator is essentially a specification of a speech act, with the propositional content of the utterance being the object of that act). The only mapping rule in this set is the following:

$$(4\text{-}48)\quad \text{KB:}\begin{bmatrix} index = X \\ spec = \begin{bmatrix} substance = F \end{bmatrix} \end{bmatrix} \Rightarrow \text{RS:}\begin{bmatrix} index = X \\ sem = \begin{bmatrix} illocforce = F \end{bmatrix} \end{bmatrix}$$

This rule establishes that the recoverable semantic structure corresponding to an eventuality which is a speech act is a structure whose illocutionary force is the speech act in question.

Mapping Rules as Path Equations

Mapping rules can also be represented as sets of PATH EQUATIONS. A PATH in a feature structure is a sequence of features which picks out some part of that structure. We notate a path by means of a sequence of symbols enclosed between angle brackets, where the first symbol is an identifier for the feature structure itself, and the second and subsequent symbols are feature names. Thus, given a structure F, the path

(4-49) $<F\ a\ b\ c>$

picks out the value of the feature c in that structure which is the value of the feature b in that structure which is the value of the feature a in the structure F.

Corresponding to the mapping rule presented above, then, we have the following set of path equations:

(4-50)
$$\begin{aligned} &<RS\ index> &&= &&<KB\ index> \\ &<RS\ sem\ illocforce> &&= &&<KB\ spec\ substance> \end{aligned}$$

In general, those path equations whose left-hand side specifies a path in the input structure and whose right-hand side specifies an atomic value are effectively conditions on the application of the rule represented by the set of path equations. That this is not made explicit in the formalism is deliberate, since it means that the equations can be read declaratively, and thus could, in theory, be used in reverse (i.e., to construct knowledge base structures from recoverable semantic structures).

In the remainder of this thesis, mapping rules will be specified as sets of path equations, since this takes less space.

Determining Tense

The propositional content (TPROP, for tensed proposition) of the recoverable semantic structure consists of two parts: the TENSE and the PROP (for proposition). Ideally, the tense of the proposition should be determined by the relative temporal indices of the speech time, event time and reference time [Reichenbach 1947]. However, we make use of a much simplified rule here, since only imperative sentences are realized in the current context:

(4-51) *<KB spec illocforce>* = *requesting*
<RS sem tprop tense> = *inf*

Choosing the Participants

In order to choose the participants to be described, for each SUBSTANCE we have a verb CASE FRAME (see, for example, [Fillmore 1968], [Bruce 1975b]) which specifies the correspondences between KB participants and RS participants. The frame for PEEL, for example, is as follows:[13]

(4-52)

PEEL	**Verb Case Role**	**Eventuality Participant**
	Object [oblig +]	In:Object

This states that the OBJECT in the recoverable semantic structure is the value of *<in object>* in the knowledge base structure. Verb case roles are marked as to whether they are OBLIGATORY: non-obligatory elements may, under circumstances to be described later, result in null NP constituents. This information can also be specified by means of path equations, thus:

(4-53) *<RS sem tprop prop participants object index>*
= <KB spec participants object participants in object index>
<RS sem tprop prop participants object spec status oblig> = +

Since the participants to be included in the recoverable semantic structure may be drawn from either the *<participants in>* or *<participants out>* features of the event to be described, the verb case frames also determine whether an entity is to be described as it was before or after the event in question. This permits us to generate sentences

[13] Again, the case frames described here are simplified in that we assume that the agent will not be explicitly mentioned in the surface linguistic form. This is acceptable in the current context since we only ever generate imperative sentences.

like

(4-54) Remove the stalks from the strawberries.

even though the stalks do not exist explicitly in the model at the outset of the action. The corresponding verb case frame in this case is

(4-55)

REMOVE	**Verb Case Role**	**Eventuality Participant**
	Object [oblig +]	Out:Result
	IndObject [oblig −]	In:Object

The output of this phase of the processing is a RS structure which might look something like the following:[14]

(4-56)

$$\left[\begin{array}{l} index = e \\ sem = \left[\begin{array}{l} illocforce = requesting \\ tprop = \left[\begin{array}{l} tense = inf \\ prop = \left[\begin{array}{l} pred = peeling \\ args = \left[\ object = \ldots\ \right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

which might be the recoverable semantics underlying the utterance

(4-57) Peel the potatoes.

However, before this sentence form can be generated, we have to construct an abstract syntactic structure. We turn to this in the next section.

4.4.3 Building the Abstract Syntactic Structure

Just as there is a one-to-many mapping from KB structures to RS structures, there is a one-to-many mapping from RS structures to AS structures: the recoverable semantics of an utterance underspecify the particular realization of that utterance. The domain of the RS→AS rules is the universe of recoverable semantic structures, and the range is the universe of abstract syntactic structures. The set of possible AS structures is defined by the following grammar:

[14] The value of the OBJECT feature here has been elided: its detail is worked out as explained in chapter 5.

(4-58)

AS	::-	index sem
index	::-	$x_i \mid e_i \mid$ *<a list of indices>*
sem	::-	mood tprop
mood	::-	*imp, decl, interr*
tprop	::-	tense prop
tense	::-	*present, past, inf* ...
prop	::-	agent action args
action	::-	*<a basic eventuality substance>*
args	::-	[object] [indobject] ...

Again, we leave the definition of the abstract syntactic structures that correspond to participants until the next chapter.

The AS structure is very close to the surface syntactic structure of the eventual utterance. For direct speech acts, there are straightforward mapping rules which determine the sentence type on the basis of the illocutionary force of the required utterance:

(4-59)
$$\begin{aligned} &<RS\ sem\ illocforce> &&= requesting \\ &<AS\ sem\ mood> &&= imp \end{aligned}$$

(4-60)
$$\begin{aligned} &<RS\ sem\ illocforce> &&= informing \\ &<AS\ sem\ mood> &&= decl \end{aligned}$$

Obviously this simple correspondence could be made more complicated by allowing indirect speech acts to be performed.

The RS→AS mapping rules also determine whether the resulting sentence will be active or passive. The fully specified AS structure for the utterance

(4-61) Peel the potatoes.

might then look like the following, where we have again omitted the structure corresponding to *the potatoes* because of lack of space:

(4-62)
$$\left[\begin{array}{l} index = e \\ sem = \left[\begin{array}{l} mood = imp \\ tprop = \left[\begin{array}{l} tense = inf \\ prop = \left[\begin{array}{l} agent = hearer \\ action = peeling \\ args = \left[\ object = \ldots\ \right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

Note that the propositional content consists of an AGENT and an ACTION, rather than a predicate and a number of arguments: thus, this intermediate representation mirrors the standard subject-predicate form of sentences more closely than ordinary first-order predicate calculus. This structure would be more useful, for example, for the generation of VP ellipsis, since it makes it very easy to compare the underlying structures corresponding to verb phrases.

Once complete, the abstract syntactic structure is then unified with the grammar to produce a surface linguistic form, as described below.

4.4.4 The Unification Grammar

Linguistically-Founded Grammars in Natural Language Generation

By and large, most work in natural language generation does not make use of linguistically-founded grammatical formalisms. A major reason for the *ad hoc* nature of grammatical knowledge encoded in these systems is that most mainstream grammatical formalisms do not represent the kinds of information required in the generation process (such as specification of focus, rhetorical force, etc.); in the absence of an existing tool with the desired specification, the researcher's response is often to grow his or her own linguistic structure building mechanisms.

There are some exceptions to this: some approaches to grammar do embody the right kinds of information. Thus, the systemic tradition (cf. Halliday [1973, 1985]) is well-represented within the field of natural language generation, and most developed in the NIGEL system (see, for example, the work by Mann [1983] and Matthiessen [1981, 1984]). Functional unification grammar (FUG) [Kay 1975] has also proved popular amongst researchers in NLG, since it too addresses functional aspects of language rather than simply being concerned with the derivation of semantic content in a narrow sense. More recently, there has been work that has tried to integrate the well-established and broad coverage of systemic grammar with the more formally robust work in FUG: see, for example, Matthiessen and Kasper [1987] and Mellish [1987]. A different direction can be found in some recent work by McDonald and Pustejovsky [1985], which uses Joshi's [1983, 1985] TREE ADJOINING GRAMMAR.

Unification now plays a central role in a number of grammar formalisms, with a bewildering variety of similar sounding names: unification categorial grammar (UCG) [Zeevat, Klein, and Calder 1987; Calder, Klein and Zeevat 1988], categorial unification grammar (CUG) [Uszkoreit 1986], dependency unification grammar [Hellwig 1986], relational unification grammar [Cohen 1984], and tree unification grammar [Popowich 1988]: doubtless there are others. Shieber [1986] provides a good introduction to the area. The extent to which these newer unification grammars have been used in NLG work is still somewhat limited, but this is unlikely to remain the case for long, since the declarative nature of the formalisms lends itself well to the development of grammars which can be used bi-directionally (see, for example, Shieber [1988]).

EPICURE's Unification Grammar

The grammatical formalism used in the present work adheres to the unification approach, and has been influenced by generalised phrase structure grammar (GPSG) [Gazdar *et al* 1985]. The mapping from the abstract syntactic structure to a surface syntactic form is achieved by unifying the abstract syntactic structure with the grammar. In an approach like FUG, this is achieved by representing the grammar by means of a feature structure, and unifying this with the representation of the semantic structure to be realized. This is the mechanism used, for example, in Appelt's TELEGRAM grammar.[15]

In EPICURE, the grammar consists of phrase structure rules annotated with path equations which determine the relationships between abstract syntactic units and surface syntactic units: the path equations specify arbitrary constituents (either complex or atomic) of feature structures. The phrase structure rules are expressed here in a PATR-like formalism [Karttunen 1986; Shieber 1986], although within EPICURE they are encoded as PROLOG definite clause grammar (DCG) rules [Clocksin and Mellish 1981].

Suppose, for example, we wish to realize the following abstract syntactic structure by means of a declarative sentence:

[15]For a detailed exposition of how this mechanism is used, see Appelt [1986:103–112].

(4-63)
$$\begin{bmatrix} index = e \\ sem = \begin{bmatrix} mood = decl \\ tprop = \begin{bmatrix} tense = past \\ prop = \begin{bmatrix} agent = j \\ action = \begin{bmatrix} pred = grate \\ object = c \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Here, j and c are standing in for more detailed representations of the participants in the event described. The corresponding surface string is then

(4-64) Judy grated some cheese.

This is an instance of the simple sentence form expressed by the phrase structure rule

(4-65) $S \rightarrow NP\ VP$

In order to generate the surface form, we annotate the phrase structure rule with path equations that determine how the elements of the abstract syntactic structure are to be distributed through the surface syntactic constituents:

(4-66) $S \rightarrow NP\ VP$

$$\begin{aligned} <NP\ Sem> &= <S\ Sem\ TProp\ Prop\ agent> \\ <VP\ Sem> &= <S\ Sem\ TProp\ Prop\ Action> \\ <VP\ Tense> &= <S\ Sem\ TProp\ Tense> \\ <NP\ Syn\ Agr> &= <VP\ Syn\ Agr> \end{aligned}$$

The abstract syntactic structure is first constrained to unify with any grammar rule whose left-hand side is S (i.e., the structure must be realised as a sentence). The path equations then cause the appropriate component structures to be passed to the rules that generate NPs and VPs, and ensure that these constituents agree syntactically:[16]

EPICURE is primarily interested in generating imperative sentences, for which we have the following rule:

(4-67) $S \rightarrow VP$

$$\begin{aligned} <S\ Sem\ TProp\ Tense> &= inf \\ <VP\ Tense> &= <S\ Sem\ TProp\ Tense> \\ <VP\ Sem> &= <S\ Sem\ TProp\ Prop\ Action> \end{aligned}$$

Here, the first constraint ensures that the rule will only unify with abstract syntactic

[16]The grammar rules sometimes contain other constraints, not shown here, to reduce inefficiency in the use of the grammar.

structures which are intended to be realized as imperatives.

4.4.5 The Clause Generation Algorithm

We are now in a position to summarise the clause generation algorithm, and its interaction with the discourse model. Given a knowledge base structure which specifies a primitive request I, the algorithm is as follows.

First, we manipulate the basic elements of the discourse model, and ensure that the world model is updated appropriately:

- Flush the previous clause contents memory.
- Move the contents of the current clause workspace into the previous clause contents memory.
- Put I into the current clause workspace.
- Pass the OBJECT of I (an eventuality specification) to the domain modeller so that it can update the world model.

EPICURE now has access to information about the state of the world both before and after the event to be described has taken place. A recoverable semantic structure corresponding to the event to be described can then be constructed, as follows:

- Apply the KB→RS mapping rules to build the basic RECOVERABLE STRUCTURE structure R from I, as detailed above.
- Retrieve the verb case frame corresponding to the SUBSTANCE of the OBJECT of I.
- For each role specified in the verb case frame do the following:
  - Get the corresponding participant x_i from I.
  - If x_i is the current CENTRE then add the attribute CENTRE with value + to this participant in R.
- Set the CENTRE to be the RESULT of the OBJECT of I.
- For each of the participants in R:

- If the participant has a CENTRE value of '+' then set the SPEC to be ϕ (i.e., no semantic content is necessary for NPs which are the focus of attention: they may be realized by means of pronouns or null NPs, as determined later).
- Otherwise, build an appropriate recoverable semantics for the noun phrase (as described in chapter 5).

The resulting RS structure is then passed to the mechanism that constructs abstract syntactic structures. This applies the RS→AS mapping rules to build a basic AS structure A, and then applies the rules described in the next chapter to construct the appropriate abstract syntactic structures for the noun phrases to be generated. The abstract syntax constructor can choose to omit entire participants, just as it can choose to omit chunks of semantic content to make use of *one*-anaphora. In particular, as we will see in the next chapter, any participant which has the features $[oblig, -]$ and $[centre, +]$ can be omitted from the abstract syntactic structure entirely.

Chapter 5

Generating Referring Expressions

In the previous chapter, we described the overall process by which EPICURE generates connected text. In this chapter, we turn to the generation of noun phrase referring expressions. Just as in the case of clause generation, there are two steps involved in this process: first we have to construct a recoverable semantic structure which represents the semantics of the noun phrase that will eventually be realized, and then from this we have to construct an abstract syntactic structure which can be passed to the grammar. Given the emphasis of the present work on referring expressions, the mapping rules used in the generation of noun phrases are considerably more complex than those used in the generation of clauses; consequently, this chapter provides a considerable amount of detail.

Section 5.1 discusses the notion of reference adopted in this thesis, and elaborates the principles we take to underly the process of referring. We also establish the particular interpretations we adopt for the terms INITIAL REFERENCE and SUBSEQUENT REFERENCE.

The remainder of the chapter describes in detail the mechanisms required for various kinds of reference. We set the scene in section 5.2 by providing an overview of the processes used in building noun phrases, exemplified by means of references to singular individuals.

In section 5.3, we describe the process of generating initial referring expressions of

various kinds, corresponding to the variety of knowledge base structures we presented in chapter 3. We show how initial references to individuals, masses and sets like the following are constructed:

(5-1) a a ring of onion
b an onion ring
c some turmeric
d two teaspoons of coriander
e some aubergines
f six courgettes
g two pounds of apples
h 350g of sultanas and raisins

In dealing with these examples, we specify most of the KB→RS and RS→AS mapping rules and most of the noun phrase grammar used within EPICURE.

In section 5.4, we turn to the question of pronominalization. We distinguish IMMEDIATE and LONG-DISTANCE PRONOMINALIZATION, and present an algorithm for immediate pronominalization that also provides for the use of NP ellipsis, as in the following example:

(5-2) a Boil the rice.
b Drain ϕ.

We then make some suggestions as to how long-distance pronominalization might be integrated into the framework used here.

Section 5.5 describes the mechanisms used in the generation of definite noun phrases, and covers a range of phenomena that arise in this connection. We introduce the notion of DISCRIMINATORY POWER as a means to determining the semantic content of a noun phrase, and show how this is used in constructing subsequent referring expressions of various kinds, including references to sets of entities and entities which have been derived from other entities, as in the following:

(5-3) a the carrots, potatoes and cucumber
b the yolk of the egg

We also make some suggestions as to how the present approach can be extended to deal with both inferrable entities and entities described in terms of events in which they have

participated, as exemplified by the following:

(5-4) a the shell
b the olives you pitted

Finally, section 5.6 looks at the mechanisms used in the generation of *one*-anaphora. Some existing approaches to this problem are described, and we suggest a solution which relies upon the distinction we draw between two intermediate levels of representation.

Having described each of the components of the process of generating referring expressions, these are collected together in section 5.7, where the complete referring expression generation algorithm is summarised.

5.1 Referring

In this section, we examine in more detail what is involved in referring to an entity.[1] We distinguish INITIAL REFERENCE and SUBSEQUENT REFERENCE, and we state some principles that govern the generation of referring expressions in EPICURE. We then summarise the kinds of information that are maintained in the hearer model.

5.1.1 Initial and Subsequent Reference

For the purposes of generating referring expressions, McDonald [1980:216] draws a distinction between initial reference and subsequent reference. These notions are intuitive enough: we use initial reference when we refer to an entity for the first time, and subsequent reference whenever an entity already mentioned is referred to again. In the present work, initial reference is the name we give to an act of reference which introduces an entity into the discourse model: thus, the point of an initial reference is to establish a symbol which can be used as a 'conceptual coathook'.[2] Subsequent reference is the name we give to an act of reference which picks out an entity which is already present in the discourse model: a speaker uses an act of subsequent reference when he or she wants to identify a discourse entity in order to say something more about it.

[1] Recall from chapter 2 that this use of the expression *refer to* is slightly misleading, since the 'referents' are machine-internal or mental entities that may or may not correspond to entities in the real world.

[2] Webber [1988] attributes the term to William Woods.

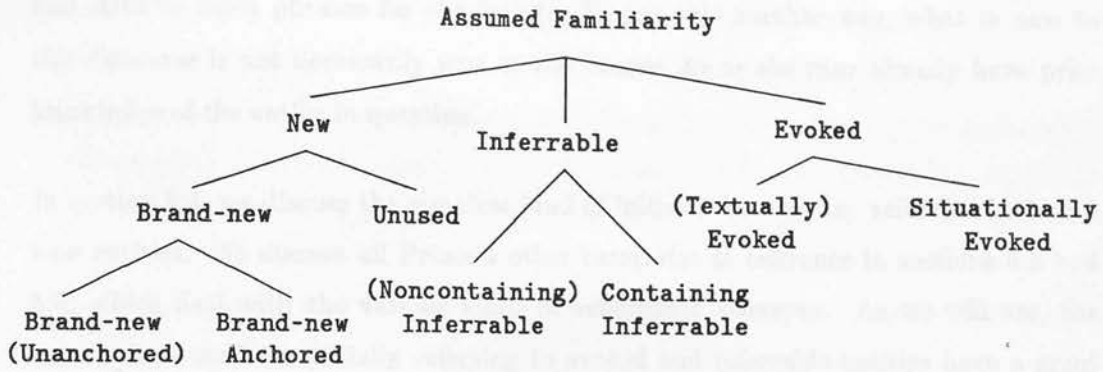


Figure 5.1: Prince's Taxonomy of Given-New Information

Prior to its being introduced into the discourse model, it is possible that the entity being referred to is already known to the hearer. An entity is known to a discourse participant if that discourse participant already has an internal symbol corresponding to that entity. If an entity is known to both discourse participants, then we say that the entity is MUTUALLY KNOWN.[3] In the present system, all entities in the domain are known to EPICURE, but not all entities are known to the hearer.

The distinction between initial and subsequent reference seems to correspond fairly straightforwardly to the normal uses of the terms *new* and *given*. However, as we saw in chapter 2, a number of intermediate levels of givenness can be hypothesized, as described by Prince's [1981:237] taxonomy of assumed familiarity, shown again in figure 5.1.

Prince's categories represent different statuses that a given entity can have with respect to a discourse. As defined above, the term 'subsequent reference' only applies to those entities which have been textually evoked, since entities falling under the other categories are not present in the discourse model when they are referred to: thus, in these terms, the other categories describe forms of initial reference. However, in terms of the kinds

[3]Mutual knowledge is a thorny problem: see, for example, Clark and Marshall [1981]. We assume a very simple model of mutual knowledge, whereby an entity is said to be mutually known if the system knows of the existence of an entity, and the system believes that the user knows of the existence of that entity.

of referring expressions used, the relevant distinction appears to be between brand-new references and all the other categories: indefinite noun phrases are used for the former and definite noun phrases for the latter. To put this another way, what is new to the discourse is not necessarily new to the hearer: he or she may already have prior knowledge of the entity in question.

In section 5.4, we discuss the simplest kind of initial reference, i.e. reference to brand new entities. We discuss all Prince's other categories of reference in sections 5.5 and 5.6, which deal with the various kinds of subsequent reference. As we will see, the mechanisms used for initially referring to evoked and inferrable entities have a great deal in common with subsequent reference to textually evoked entities, and so it makes sense to treat them together.

5.1.2 The Principles of Reference

We can view the process of constructing a referring expression as being governed by a number of principles, very like Gricean conversational maxims [Grice 1975], which we will call the principles of SENSITIVITY, EFFICIENCY and ADEQUACY.

The Principle of Sensitivity

A speaker should always be *sensitive* to her hearers. The principle of sensitivity requires that, in constructing a referring expression, the speaker must pay heed to what the hearer can be presumed to know. At the broadest level, this gives rise to requirements like:

- use a language that the hearer understands; and
- use words and expressions that the hearer understands.

The principle also requires that the speaker should acknowledge when an entity being referred to is already known to the hearer, since to do otherwise may be misleading. Thus, the principle requires that definite and indefinite referring expressions should be used as appropriate.

The Principle of Adequacy

A referring expression should identify the intended referent unambiguously, and provide sufficient information to serve the purpose of the reference. Thus, if there are two balls on a table, and I wish you to pick up a particular one, then referring to it as *the ball* is unlikely to be adequate: I must distinguish the particular ball I have in mind from the other ball, perhaps by making reference to its colour, size or location.

The adequacy of an act of reference is dependent upon the purpose of the reference: a referring expression which is sufficient to permit the hearer to construct or retrieve a mental entity corresponding to the intended referent may not be adequate for actually locating the referent in the real world. Given the latter purpose, if there is only one tin of cat food in the kitchen and I say to you

(5-5) Bring me the tin of cat food.

then the referring expression I have used will only be adequate if I have provided sufficient information to find the tin of cat food in question within the referring expression itself. If the tin of cat food is hidden in an unusual location, then a referring expression like that in the request

(5-6) Bring me the tin of cat food which is under the bucket on the top shelf.

may be necessary in order to satisfy the principle of adequacy.

The Principle of Efficiency

The principle of efficiency requires that the referring expression used must not contain more information than is necessary for the task at hand: as such, it pulls in the opposite direction from the principle of adequacy. This is not just a question of the speaker saving her breath; saying more than is necessary is likely to mislead the hearer into thinking that something else is encoded in the message. Satisfying this principle requires some way of measuring the amount of information in a referring expression: we will return to this question in section 5.5, where we introduce the notion of DISCRIMINATORY POWER.

5.1.3 The Hearer Model

We noted in chapter 4 one particular kind of knowledge about the hearer that the speaker needed to have in order to be able to plan a discourse: namely, knowledge about the hearer's capabilities with respect to the actions in plans. From the above discussion, we can identify other kinds of knowledge of the hearer that the speaker must have:

1. the speaker must know what language the hearer understands;
2. the speaker must know what words and expressions in that language the hearer understands;
3. the speaker must know which entities are known to the hearer by virtue of having been explicitly mentioned in the discourse;
4. the speaker must know which entities are known to the hearer, besides those which have been explicitly mentioned in the discourse;
5. the speaker must know what entities the hearer can infer the existence of; and
6. the speaker must know what entities are assumed by the hearer to be, in Prince's terms, situationally evoked.

In the present system, we ignore the problems presented by (1) and (2) above: we assume that the hearer understands English, and that she understands all the vocabulary available to EPICURE.[4]

In the case of (3) above, the system's knowledge of which entities are known to the hearer by virtue of having been explicitly mentioned is modelled by the shared discourse model introduced in chapter 4.

In the case of (4) above, entities which are known to the hearer, but which have not been mentioned in the discourse, are present in the hearer model by virtue of assertions in EPICURE's knowledge base of the following form:

(5-7) Knows(hearer, holds($s, A(x,V)$))

[4]This is not strictly true: the system's knowledge of what actions the hearer is capable of performing, as described in chapter 4, embodies a limited model of the names of cooking operations that the user understands.

where x is some entity, A is an arbitrary attribute of that entity, and V is the value of A for x. Thus, if the hearer knows anything at all about an entity x, then x is known to the hearer.

In order to cater for entities whose existence can be inferred (case (5)), we have assertions in EPICURE's knowledge base regarding the axioms known to the hearer.[5] So, if the hearer knows that corresponding to every avocado there is a stone, and EPICURE knows that the hearer knows this, then we have:

(5-8) Knows(hearer, $[\forall x \text{ avocado}(x) \supset \exists y \text{ stone}(y) \wedge \text{associated-with}(y, x)]$)

Finally, to deal with case (6), we have assertions regarding those entities which are situationally evoked. In the present work, only the speaker and hearer are considered to be situationally evoked:

(5-9) Knows(hearer, holds(s, situationally-evoked(speaker)))
Knows(hearer, holds(s, situationally-evoked(hearer)))

The effect of this information is that the system can use pronouns to refer to itself and the user.

In the remainder of this chapter, we show how this knowledge of the hearer's knowledge, in conjunction with the discourse model, results in the generation of referring expressions that conform to the principles of reference described above.

5.2 Building Noun Phrases

In this section, we describe the various stages of processing required in order to construct noun phrases. The most important aspect of this process is that of deciding on the semantic content to be realised in the noun phrase. The emphasis in this section is on describing what happens at the various stages of processing, and how information is transferred between the different levels of representation: as before, this is achieved by the use of MAPPING RULES. We describe a number of the mapping rules, but leave the question of how EPICURE decides *which* rules to use until later in the present chapter. As an example of how the mechanism operates, we will consider references to singular

[5]The mechanisms required to use axioms like these are discussed later in this chapter, and are currently being implemented, but are not present in the version of the system described in chapter 6.

individuals.

5.2.1 An Overview of the Process

Just as in the case of generating clauses as a whole, we distinguish several levels of representation in the generation of a noun phrase:

- the knowledge base object corresponding to the entity to be described;
- the recoverable semantic structure chosen for the noun phrase that will describe that entity;
- the abstract syntactic structure of the noun phrase that will describe that entity; and
- the surface syntactic structure of the resulting noun phrase.

Again, just as in the case of clause generation, we have two sets of mapping rules that create the two intermediate levels of structure. However, in the case of the generation of noun phrase referring expressions, these are considerably more complex than the rules used in clause generation. In building the recoverable semantic structure, the system has to decide what the hearer should understand the noun phrase to mean once any anaphoric elements have been resolved. This involves choosing the particular KB→RS mapping rules that will have the desired effect. Constructing the abstract syntactic structure is then relatively straightforward, since the applicability of RS→AS rules is tightly constrained by the recoverable semantic structure. This stage of the process is non-deterministic, however, since different rules may apply to one recoverable semantic structure, producing different results. Thus, for example, the two noun phrases

(5-10) a a Glasgow-bound plane
b a plane to Glasgow

have the same recoverable semantic structure, but different abstract syntactic structures.

In order to demonstrate the various stages of constructing a noun phrase referring expression, we consider below the construction of simple indefinite and definite referring expressions like the following:

(5-11) a a mouldy carrot

b the carrot

These two referring expressions represent an initial and a subsequent reference to a singular individual; we deal with references of both kinds to more complex entities in the remainder of the chapter. In each case, the stages of processing involved are as follows:

- given a knowledge base structure, EPICURE has to construct a recoverable semantic structure that contains the information to be conveyed to the hearer;
- once a recoverable semantic structure has been constructed, EPICURE then constructs an abstract syntactic structure that specifies what is to be realized in the noun phrase;
- unification of this structure with the grammar and lexicon then results in a surface string that realizes the required semantics.

We describe each of these stages in turn.

5.2.2 Determining the Recoverable Semantics

Given a knowledge base structure, a recoverable semantic structure is constructed by applying a set of KB→RS mapping rules. The recoverable semantic structure has to represent two kinds of information about the intended referent:

- information which indicates the DISCOURSE STATUS of the entity; and
- a description of the entity.

A grammar that defines the possible set of recoverable semantic structures is presented at the end of this section; for the moment, we present a fragment of the recoverable semantics grammar that is relevant to the current example:

(5-12)

RS	::-	index sem
index	::-	x_i
sem	::-	status spec
status	::-	given [unique]
given	::-	+|−
unique	::-	+|−
spec	::-	agr type
agr	::-	[number] countable
countable	::-	+|−
number	::-	$sg\|pl\|1 \ldots n$
type	::-	category [props]
category	::-	*<a basic semantic category>*
props	::-	$<a_1, v_1>, \ldots, <a_n, v_n>$

The particular rules required in order to derive the information to be encoded in a referring expression depend on the type of knowledge base structure to be described, and on some contextual factors. Initial RS structures are created from KB structures by the following mapping rule:

(5-13) $<RS\ index> = <KB\ index>$

Below, we show how different information is added to the basic RS structure created by this rule, depending upon whether an initial or subsequent reference is being made.

Discourse Status

As we saw in chapter 2, determiners in English specify two things:

- whether the entity described is considered to be mutually known; and
- whether the description supplied is uniquely true of the intended referent in the relevant context.

This information is specified in the recoverable semantics by the STATUS features. We will return to how the values of the STATUS features are actually determined in subsequent sections of this chapter; for the moment, it is sufficient to note that, when a brand new entity is introduced into the discourse, the value of the STATUS feature in the corresponding recoverable semantic structure is as follows:

(5-14) $\left[\ given = - \ \right]$

Whenever an entity is subsequently referred to, however, the value of the GIVEN feature in the corresponding recoverable semantic structure will be +. If the semantic content of the description constructed to refer to that entity picks it out uniquely (by means of a process described later), then this is also noted in the STATUS feature, resulting in the following structure:

(5-15) $\begin{bmatrix} given = + \\ unique = + \end{bmatrix}$

As we will see below, these two structures result in indefinite and definite determiners respectively. If a unique description cannot be constructed, then the following structure is set to be value of the STATUS structure:

(5-16) $\begin{bmatrix} given = + \\ unique = - \end{bmatrix}$

This then also results in the use of an indefinite determiner.

Recoverable Semantic Content

Different rules are applied to build the recoverable semantic structure depending upon the value of the KB structure's STRUCTURE feature. For the moment, we will focus on objects whose structure is INDIVIDUAL. The first rule to be applied adds the appropriate agreement features to the recoverable semantic structure:

(5-17)
$$\begin{aligned} &<KB\ spec\ structure> &&= individual \\ &<RS\ sem\ spec\ agr\ countable> &&= + \\ &<RS\ sem\ spec\ agr\ number> &&= sg \end{aligned}$$

This ensures that the entity will be described as a singular individual.

Next, the semantic content of the description is determined. For objects whose structure is *individual*, the important information here are the object's SUBSTANCE, and the SHAPE of its PACKAGING. If these are the same (i.e., the entity is packaged in the 'standard way' for its substance), then the following rule applies:

(5-18)
$$\begin{aligned} &<KB\ spec\ substance> &&= <KB\ spec\ packaging\ shape> \\ &<RS\ sem\ spec\ type\ category> &&= <KB\ spec\ substance> \\ &<RS\ sem\ spec\ type\ props\ size> &&= <KB\ spec\ packaging\ size> \end{aligned}$$

Finally, other properties of the object may be added to the RS structure. Whether this is done, and which properties are added, depends on the context, and is discussed further in subsequent sections of this chapter. The addition of other properties to the recoverable semantic structure is performed by a set of rules of the form:[6]

(5-19) $<RS\ sem\ spec\ type\ props\ A> \ = \ <KB\ spec\ A>$

where the possible range of values for A is restricted (in particular, the SUBSTANCE or STRUCTURE of an entity will not be added as a property). The particular values of A used are chosen on the basis of a notion of discriminatory power, as described in section 5.4.

Suppose we have the following knowledge base structure, which represents a single mouldy carrot:

(5-20) KB =
$$\begin{bmatrix} index = x \\ state = s \\ spec = \begin{bmatrix} substance = carrot \\ structure = individual \\ packaging = \begin{bmatrix} shape = carrot \\ size = regular \end{bmatrix} \\ mouldy = + \end{bmatrix} \end{bmatrix}$$

The recoverable semantic structure corresponding to an initial reference to our carrot might then be as follows:

(5-21) RS =
$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = - \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = sg \end{bmatrix} \\ type = \begin{bmatrix} category = carrot \\ props = \begin{bmatrix} size = regular \\ mouldy = + \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

[6]Note that (5-19) is effectively a rule schema. In the system itself, we have a distinct rule for each property.

The entity's property of being mouldy has been added here by an instance of the general property addition rule we specified above, namely:

(5-22) $<RS\ sem\ spec\ type\ props\ mouldy> = <KB\ spec\ mouldy>$

However, when making a subsequent reference to the entity, the carrot's being mouldy might be irrelevant; the RS structure would then be as follows:

(5-23)
$$\text{RS} = \begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = + \\ unique = + \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = sg \end{bmatrix} \\ type = \begin{bmatrix} category = carrot \\ props = \begin{bmatrix} size = regular \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

5.2.3 Determining the Abstract Syntactic Structure

The recoverable semantic structure above cannot be realised directly by a noun phrase: first, we must build an abstract syntactic structure. The abstract syntactic structures relevant to the present example are defined by the following grammar fragment (again, a complete specification of abstract syntactic structures will be presented at the end of this section):

(5-24)

sign	::-	index phon sem syn
index	::-	x_i
phon	::-	*<a string>*
syn	::-	*<a representation of the expression's syntax>*
sem	::-	status spec
status	::-	given [unique]
given	::-	+|−
unique	::-	+|−
spec	::-	agr desc
agr	::-	countable [number]
countable	::-	+|−
number	::-	*sg*|*pl*|1 . . . *n*
desc	::-	head [mod]
head	::-	*<an atomic semantic category>*|mod head
mod	::-	head [mod]

Recall that the principal differences between the two levels of representation are that (a) the abstract syntactic structure is closer in structure to the surface syntax, and (b) some information present in the recoverable semantic structure may be omitted from the abstract syntactic structure. In the present example, the RS→AS mapping rules dealing with STATUS and AGR information are very straightforward:

(5-25)

<AS index>	=	*<RS index>*
<AS sem status>	=	*<RS sem status>*
<AS sem spec agr>	=	*<RS sem spec agr>*

In all the examples dealt with in the present work, we assume that the semantic number of an entity is the same as its syntactic number. However, the indirection provided by the use of two levels of representation would permit us to introduce a distinction between the two kinds of number, thus allowing us to cater for the representation of entities whose syntactic number seems at odds with their 'conceptual' number. Thus, in the case of noun phrases like

(5-26) a the trousers
b the pliers

the semantic number of the entities so described would be singular, whereas the syntactic number is plural.

As just noted, a major difference in the two levels of representation is the way in which the various properties to be realised are structured. In the case of the recoverable semantic structure, we isolate the semantic category that will eventually be realised as the head noun, and maintain all the other properties essentially as a flat list; in the abstract syntactic structure, however, the various properties are collected together in a way that mirrors exactly the embedding of modifiers in the noun phrase. The structure of the information in the recoverable semantics is, as we will see in section 5.6, particularly suitable for decisions concerning the use of *one*-anaphoric expressions.[7]

In order to build this part of the abstract syntactic structure, then, the generator has to know something about adjective ordering. The psychological data suggests that certain adjective orderings are preferred [Hermann and Laucht 1976], although these orderings can easily be overridden (see, for example, Ney [1983]). The most comprehensive discussion of adjective ordering is that presented by Vendler [1968], who groups adjectives into nine major categories, with one of these categories being further subcategorised into 14 groups. Following Vendler, we annotate the major categories here as A_1 through A_9, with the subcategories of A_1 bearing alphabetic indices: some examples of these categories are as follows.

A_9: *probably*, *likely*, *certain*

A_8: *useful*, *profitable*, *necessary*

A_7: *possible*, *impossible*

A_6: *clever*, *stupid*, *reasonable*, *nice*, *kind*, *thoughtful*, *considerate*

A_5: *ready*, *willing*, *anxious*

A_4: *easy*

A_3: *slow*, *fast*, *good*, *bad*, *weak*, *careful*, *beautiful*

A_2: contrastive/polar adjectives: *long-short*, *thick-thin*, *big-little*, *wide-narrow*

A_j: verb-derivatives: *washed*

A_i: verb-derivatives: *washing*

A_h: *luminous*

A_g: *rectangular*

[7] The same consideration led to Webber's [1979] use of restricted quantification in her semantic representation, so that the property corresponding to the head noun in a noun phrase would be separated out from the other properties of the entity described by the noun phrase.

A_f: colour adjectives

A_a: *iron*, *steel*, *metal*

When realized in noun phrases, according to Vendler, adjectives are ordered in the following way:

(5-27) $A_9\ A_8\ A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_x\ A_m\ A_l\ A_k\ A_j\ A_i\ A_h\ A_g\ A_f\ A_e\ A_d\ A_c\ A_b\ A_a$

Within EPICURE, each adjective is given a numerical classification as suggested by Vendler's categorisation.

Recall that the properties predicated of an entity appear as the value of the PROPS feature in the recoverable semantic structure. This is essentially a list of attribute-value pairs of the form

(5-28) $[<a_1,\ v_1>,\ <a_2,\ v_2>,\ \ldots,\ <a_n,\ v_n>]$

In order to integrate this information into the abstract syntactic structure, we have to do two things:

- omit any properties which the hearer should be able to infer; and
- embed the remaining properties in such a way that their ordering matches that specified by the adjective categorisation.

The hearer is assumed to be able to infer those properties which are defaults for entities of the specified SUBSTANCE and PACKAGING: so, for example, *regular* is assumed to be the default size of most entities.

The algorithm for integrating properties into the AS structure is as follows. First, we construct a list L of the properties to appear in the abstract syntactic structure, determining the ordering category for the adjective which realizes the semantic content of each property to be realized, while also omitting those properties whose values can be assumed by the hearer.[8]

For each $<a_i,\ v_i>$:

[8]Note that this means we assume that, if a given property is realizable by means of a number of different adjectives, then all those adjectives will belong to the same category.

- If v_i is the default value of a_i for entities of the specified SUBSTANCE and PACKAGING, then ignore this $<a_i, v_i>$ pair.
- Otherwise, determine the ordering category A_i of a_i, and add the pair $<A_i, <a_i, v_i>\!>$ to the list L.

So, for example, if an entity which is an onion has the size *regular*, this information will not be included in the list of properties to be included in the abstract syntactic structure.

Next, we have to build the abstract syntactic structure. Suppose this is a feature structure with the name S: we use the list of annotated properties L to build S recursively as follows.

- If L is empty, then set $<S\ head>$ to be the value of $<RS\ sem\ spec\ type\ category>$.
- Otherwise:
  - Find the highest-valued A_i in L (this corresponds to the adjective which will be realized furthest away from the head noun).
  - Set $<S\ mod\ head>$ to be the value of $<RS\ sem\ spec\ type\ props\ a_i>$.
  - Remove $<A_i, <a_i, v_i>\!>$ from L, and build $<S\ head>$ using the new L.

Thus, we pick off the properties to be realized from L, building up the abstract syntactic structure as we go. The result of this process is a feature structure which then becomes the value of $<AS\ sem\ spec\ desc>$. For our initial reference to our mouldy carrot, the following abstract syntactic structure results:

(5-29) AS =
$$\left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[\ given = -\ \right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} countable = + \\ number = sg \end{array}\right] \\ desc = \left[\begin{array}{l} head = carrot \\ mod = \left[\ head = mouldy\ \right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

The abstract syntactic structure for the subsequent reference, however, includes no additional properties, and so is as follows:

(5-30) AS =
$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = + \\ unique = + \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = sg \end{bmatrix} \\ desc = \begin{bmatrix} head = carrot \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Note the content of the STATUS feature here: we assume that, for the purposes of this example, the content of the DESC feature is sufficient to identify the intended referent, and so the UNIQUE feature has the value +.

5.2.4 Surface Realization

Finally, we have to realise these abstract syntactic structures as surface linguistic forms. We analyse the surface syntactic structure of

(5-31) the carrot

by the phrase structure rules

(5-32) $NP \rightarrow Det\ N1$
$N1 \rightarrow N$

resulting in the structure shown in figure 5.2; for noun phrases like

(5-33) an old carrot

we have the additional phrase structure rules

(5-34) $N1 \rightarrow AP\ N1$
$AP \rightarrow Adv\ AP$
$AP \rightarrow A$

resulting in the structural analysis shown in figure 5.3. Since the phrase structure rule that introduces adjectival phrases is recursive, the noun phrase

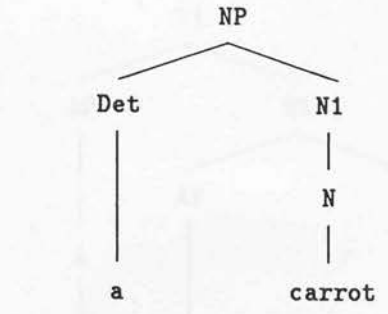


Figure 5.2: A Simple Noun Phrase

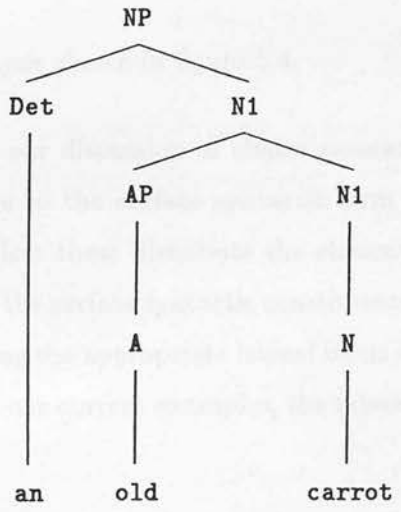


Figure 5.3: Simple Adjectival Modification

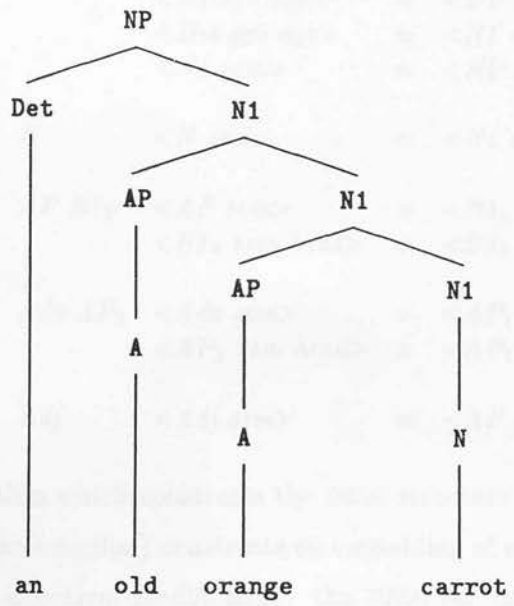


Figure 5.4: Embedded Adjectival Modification

(5-35) an old orange carrot

is given the structural analysis shown in figure 5.4.

As we saw in chapter 4 in our discussion of clause generation, the mapping from the abstract syntactic structure to the surface syntactic form is achieved by annotations on the phrase structure rules: these distribute the elements of the abstract syntactic structures appropriately to the surface syntactic constituents. The surface string is then constructed by concatenating the appropriate lexical items in the order specified by the phrase structure rules. For our current examples, the relevant rules are as follows:

(5-36)

NP	→	*Det N1*	*<Det sem>*	=	*<NP sem status>*
			<NP syn agr>	=	*<NP sem spec agr>*
			<N1 syn agr>	=	*<NP syn agr>*
			<Det syn agr>	=	*<N1 syn agr>*
			<N1 sem>	=	*<NP sem spec desc>*
N1	→	*N*	*<N sem>*	=	*<N1 sem head>*
$N1_1$	→	*AP* $N1_2$	*<AP sem>*	=	$<N1_1$ *sem mod>*
			$<N1_2$ *sem head>*	=	$<N1_1$ *sem head>*
AP_1	→	*Adv* AP_2	*<Adv sem>*	=	$<AP_1$ *sem mod>*
			$<AP_2$ *sem head>*	=	$<AP_1$ *sem head>*
AP	→	*Adj*	*<Adj sem>*	=	*<AP sem head>*

Recall that the algorithm which constructs the DESC structure in the abstract syntax (described in the previous section) constructs an embedding of modifiers corresponding to the embedding of adjectival modification: the DESC for the noun phrase *the old orange carrot* is then as shown in (5-37).

(5-37)
$$\left[desc = \left[\begin{array}{l} mod = \left[head = old \right] \\ head = \left[\begin{array}{l} mod = \left[head = orange \right] \\ head = carrot \end{array} \right] \end{array} \right] \right]$$

Mirroring the fact that adjectives themselves can be modified by adverbs, modifiers can also be modified in the abstract syntactic structure:

(5-38)
$$\left[desc = \left[\begin{array}{l} mod = \left[head = \left[\begin{array}{l} mod = \left[head = very \right] \\ head = old \end{array} \right] \right] \\ head = \left[\begin{array}{l} mod = \left[head = orange \right] \\ head = carrot \end{array} \right] \end{array} \right] \right]$$

Accordingly, the noun phrase

(5-39) the very old orange carrot

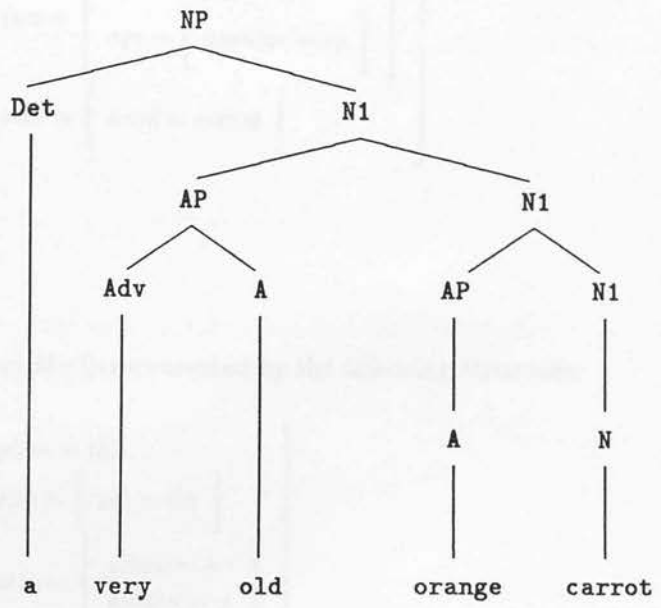


Figure 5.5: Complex Embedded Adjectival Modification

is given the surface syntactic analysis shown in figure 5.5.

5.2.5 Lexical Realization

Terminal categories in grammar rules correspond to the values of the $<$*syn cat*$>$ features of structures held in the lexicon. Thus, lexical insertion involves obtaining the lexical item with the appropriate semantics and syntactic category from the lexicon. Below, we give some examples of lexical items.

Nouns

Common nouns are represented in a fairly simple manner; for example, the noun corresponding to the semantic category *carrot* is as follows.

(5-40)
$$\begin{bmatrix} phon = carrot \\ syn = \begin{bmatrix} cat = noun \\ agr = \begin{bmatrix} number = sg \end{bmatrix} \end{bmatrix} \\ sem = \begin{bmatrix} head = carrot \end{bmatrix} \end{bmatrix}$$

Determiners

The lexical item *the* is represented by the following structure:

(5-41)
$$\begin{bmatrix} phon = the \\ syn = \begin{bmatrix} cat = det \end{bmatrix} \\ sem = \begin{bmatrix} given = + \\ unique = + \end{bmatrix} \end{bmatrix}$$

Notice that no number agreement features are specified here, so that the determiner can be used for both singular and plural noun phrases.

The indefinite determiner *a* has the following lexical entry:

(5-42)
$$\begin{bmatrix} phon = a \\ syn = \begin{bmatrix} cat = det \\ agr = \begin{bmatrix} number = sg \end{bmatrix} \end{bmatrix} \\ sem = \begin{bmatrix} given = - \end{bmatrix} \end{bmatrix}$$

In this entry, there is no specification for the UNIQUE feature, since whether or not the description used in conjunction with the determiner is unique is irrelevant.

Recall that the determiner *a* can also be used in non-specific references. In the present framework, this would require a distinct lexical entry.

5.2.6 The Levels of Representation Summarised

Above, we focused on some relatively simple semantic structures. We present here the complete specifications of the two levels of representation.

Recoverable Semantic Structure

The structure of RS objects corresponding to participants in described eventualities is described by the following grammar:

(5-43)

RS	::-	index sem
index	::-	x_i|<*a list of indices*>
sem	::-	status spec
status	::-	given [unique]
given	::-	+|−
unique	::-	+|−
spec	::-	agr type|
		agr quant subst|
		agr part set
agr	::-	[number] countable
type	::-	category [props]
category	::-	<*a basic semantic category*>
props	::-	$<a_1, v_1>, \ldots, <a_n, v_n>$
quant	::-	agr type
subst	::-	agr type|<*a list of subst structures*>
part	::-	agr type
set	::-	index sem

Abstract Syntactic Structure

The abstract syntax is similar in a number of respects to the recoverable semantics, and is described by the following grammar.

(5-44)

sign	::-	index phon sem syn
index	::-	$x_i \mid e_i \mid$ *<a list of indices>*
phon	::-	*<a string>*
syn	::-	*<a graph representation of the expression's syntax>*
sem	::-	status spec
status	::-	given [unique]
given	::-	$+ \mid -$
unique	::-	$+ \mid -$
spec	::-	agr desc|*<a list of spec structures>*
agr	::-	countable [number]
countable	::-	$+ \mid -$
number	::-	$sg \mid pl \mid 1 \ldots n$
desc	::-	head [mod]|spec_1 spec_2|spec set
head	::-	*<an atomic semantic category>*|mod head
mod	::-	head [mod]
set	::-	index phon sem syn

The remainder of this chapter will describe the mapping rules used in building the wider range of structures defined by these grammars.

5.3 Initial Reference

Within the literature on natural language generation, there is very little work which addresses the issues involved in initial reference in any great depth. The main problem is that of deciding what the semantic content of an initial reference should be. In the present work, we adopt a fairly simple solution which is adequate in the domain addressed in this thesis, but at the same time permits us to generate a wide range of surface forms for initial reference, corresponding to the wide variety of objects we saw in chapter 3. Below, we elaborate on the problematic nature of initial reference. We then go on to describe the basic mechanisms used in EPICURE for the construction of a variety of initial references.

5.3.1 The Problem of Initial Reference

Deciding on the content of an initial reference to an entity is not a simple process. We might expect things to be simplest when the intended referent has a proper name that is known to both the speaker and the hearer; but even in this case there are subtle decisions that have to be made on the basis of contextual factors, as McDonald points out:

> When I telephone one of my house-mates at their laboratory, context comes into play in deciding which name I use to refer to them. If I recognise who answers, I say "Is Jeanne there?". I use their first name because it is the friendly thing to do and because I'm sure they will know who I want. If I don't recognise them, then I use the full name, "Is Jeanne Margolskee there?". That is more formal and less likely to be confusing. Before I knew that there was no one else in that lab with the same first name, I always used the full name. If I should want to sound official, I would probably say "Is Dr. Margolskee there?".
>
> [McDonald 1977:119]

If the intended referent does not have a mutually-known name (and this is generally the case) then it will be described in terms of its properties; and which properties we use again depends very much on the context of use and the purpose of making the reference in the first place.

As we suggested at the end of chapter 3, descriptions of a given object are always mediated through a PERSPECTIVE: a crucial issue, then, when referring to an entity for the first time, is the choice of perspective from which the entity is to be viewed. When referring to a friend's mother, for example, I have to decide whether to view her (for the purposes of the conversation) as an architect, a mother, or a mountain climber. The mechanisms underlying this choice are ill-understood, and essentially unexplored within computational linguistics: this is hardly surprising, given that all the systems extant at the time of writing operate within narrowly defined domains where perspective choice is unnecessary, with one perspective being built-in to the system. The system described in this thesis is no less deficient, since we assume that the entities within recipes are to be described as ingredients. This simplification does no harm in the present work.

Once the initial perspective to be taken has been determined, the search space of properties that can be used in describing an entity for the first time is reduced considerably. The purpose of the discourse remains a major factor in the particular choices made, however. For example, if we are talking about the variety of hobbies pursued by our friend's mothers, then the property of being a mountain climber might be the only relevant property I have to mention; but if we are talking about quality of mountain climbing skill exhibited by people we know, I might introduce my friend's mother as *a very good mountain climber*. In general, the requirement seems to be to inform the hearer of those properties which are necessary to distinguish the intended referent from

other entities that have been mentioned, and often (given sufficient prescience on the part of the speaker) those properties which are necessary to distinguish it from entities the speaker expects to be introduced later in the discourse.

5.3.2 Initial Reference in Epicure

Within EPICURE, there are two requirements that guide the choice of the properties that are used in describing an ingredient for the first time:

- sufficient information should be provided to distinguish the intended referent from all other entities that have already been mentioned, and also those that have yet to be mentioned;
- those specific properties which are relevant in the context of cooking (namely substance, quantity, and any prior processing that is required) should be mentioned.

Recall that, at the outset of a description of a recipe, all the ingredients are elements of the WORKING SET. Each element of the working set is described by constructing a noun phrase that encodes the relevant information held in the knowledge base structure that represents this ingredient. In the remainder of this section, we show the various processes involved: as before, we first construct a recoverable semantic structure, and then develop from this an abstract syntactic structure which can be realized by the grammar.

As mentioned in section 5.2, information about the discourse status of the entity being described, and the uniqueness-in-context of the description used, determine the contents of the STATUS feature which is introduced in the recoverable semantic structure, and then copied across to the abstract syntactic structure. For entities which are being referred to for the first time, the relevant status information is specified by the following mapping rule:

(5-45) $<RS\ spec\ status\ given> = -$

This rule applies whenever the entity to be referred to is brand-new; i.e., provided the intended referent is not present in the discourse model, is unknown to the hearer, and its existence cannot be inferred by the hearer. In the RS structures described below, we

will assume that this feature has already been added. We now consider the construction of initial references to individuals, masses and various kinds of sets that can appear in the domain.

5.3.3 Initial Reference to Individuals

The various steps involved in constructing an initial reference to an entity whose STRUCTURE is *individual* were described in section 5.2; the only thing not specified there was the algorithm which determines the particular properties to be used. This is very straightforward:

If the STRUCTURE of an entity x is *individual*:

- if x has an ancestor y then include a description of y; otherwise, describe the SUBSTANCE of x;[9]
- describe the PACKAGING of x; and
- describe any other locally-encoded properties of x.

A LOCALLY-ENCODED PROPERTY is any property of the entity, other than its STRUCTURE, SUBSTANCE, PACKAGING, and ANCESTOR, which is represented as part of the entity's SPEC feature. The restriction to locally-encoded properties prevents EPICURE from including, for example, information which might be inherited from other entities (such as, for example, the information that the entity in question is a member of the working set).

The procedure described in section 5.2 is therefore all that is involved for simple individuals as might be described by the following noun phrases:

(5-46) a a carrot
b a ripe green pepper

However, there are two other kinds of individuals which are slightly more complex. Firstly, we have entities whose PACKAGING is not the 'standard' packaging for entities made of that substance, as in

[9]Recall from chapter 3 that an entity may be represented as being derived from another entity, where that other entity is said to be the ANCESTOR of the first.

(5-47) a a ring of onion
b a stick of celery

Secondly, we have entities which are described in terms of other entities from which they have been derived (their ANCESTORs), as in

(5-48) a the juice of a lemon
b the kernels of a fresh ear of corn

We will return to the second category later in the present chapter, since the mechanisms used are closer to those used for certain kinds of subsequent reference. Here, we show how the system constructs initial references to entities, like that described by *a ring of onion*, which belong to the first category.

Building the Recoverable Semantics

Recall from chapter 3 that the knowledge base structure corresponding to an entity describable as *a ring of onion* will look like the following:

(5-49) KB =
$$\left[\begin{array}{l} index = x \\ state = s \\ spec = \left[\begin{array}{l} structure = individual \\ substance = onion \\ packaging = \left[\begin{array}{l} shape = ring \\ size = regular \end{array}\right] \end{array}\right] \end{array}\right]$$

For entities of this kind, where the SUBSTANCE and the SHAPE of PACKAGING are not the same, we have the following mapping rule:

(5-50) $$\begin{array}{lcl} <RS\ sem\ spec\ type\ category> & = & <KB\ spec\ packaging\ shape> \\ <RS\ sem\ spec\ type\ props\ substance> & = & <KB\ spec\ substance> \\ <RS\ sem\ spec\ type\ props\ size> & = & <KB\ spec\ packaging\ size> \end{array}$$

The resulting recoverable semantic structure is then

(5-51)
$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = - \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = sg \end{bmatrix} \\ type = \begin{bmatrix} category = ring \\ props = \begin{bmatrix} size = regular \\ substance = onion \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Building the Abstract Syntactic Structure

The values for $<AS\ sem\ status>$ and $<AS\ sem\ spec\ agr>$ are the same as the STATUS and AGR features in the RS structure, and are specified by the mapping rules already presented. However, note that entities of this kind can be described in two ways:

(5-52) a a ring of onion
b an onion ring

In order to account for this, we have two distinct abstract syntactic structures that correspond to the recoverable semantics, and therefore two sets of mapping rules. These can apply whenever the SUBSTANCE of an entity is specified as one of the PROPS in the recoverable semantics.

In order to generate noun phrases like (5-52a), we require an abstract syntactic structure whose SPEC feature is of the following general form:

(5-53)
$$\begin{bmatrix} spec = \begin{bmatrix} agr = \ldots \end{bmatrix} \\ desc = \begin{bmatrix} spec_1 = \begin{bmatrix} agr = \ldots \\ desc = \ldots \end{bmatrix} \\ spec_2 = \begin{bmatrix} agr = \ldots \\ desc = \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where the embedded SPEC_1 feature corresponds to the packaging of the object, and the embedded SPEC_2 feature corresponds to the substance of the object. The RS→SS

mapping rules which build this structure are as follows, where R is the value of $<RS\ sem\ spec>$ and A is the value of $<AS\ sem\ spec\ desc>$:

(5-54)
$$\begin{array}{lcl} <A\ spec_1\ desc\ head> & = & <R\ type\ category> \\ <A\ spec_1\ agr> & = & <A\ agr> \\ <A\ spec_2\ desc\ head> & = & <R\ type\ props\ substance> \\ <A\ spec_2\ agr\ countable> & = & - \\ <A\ spec_2\ agr\ number> & = & sg \end{array}$$

The resulting abstract syntactic structure is then

(5-55)
$$\left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[\ given = -\ \right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = sg \\ countable = + \end{array}\right] \\ desc = \left[\begin{array}{l} spec_1 = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = sg \\ countable = + \end{array}\right] \\ desc = \left[\ head = ring\ \right] \end{array}\right] \\ spec_2 = \left[\begin{array}{l} agr = \left[\ countable = -\ \right] \\ desc = \left[\ head = onion\ \right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

In order to construct the abstract syntactic structure corresponding to *an onion ring*, we have a different set of mapping rules:

(5-56)
$$\begin{array}{lcl} <A\ head> & = & <R\ type\ head> \\ <A\ mod\ head> & = & <R\ type\ props\ substance> \end{array}$$

Thus, in this case, the abstract syntactic structure is as follows:

(5-57) AS =
$$\left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[given = - \right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = sg \\ countable = + \end{array}\right] \\ desc = \left[\begin{array}{l} head = ring \\ mod = \left[head = onion \right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

Building the Surface Syntactic Structure

The syntactic analyses we provided for the simple noun phrases we presented in section 5.2 were relatively uncontroversial. There is much less agreement within the linguistics literature, however, as to the correct analysis of noun phrases like *a ring of onion* (see, for example, Akmajian and Lehrer [1976] and Selkirk [1977]).

Constructions like these are usually referred to as PSEUDO-PARTITIVES [Selkirk 1977:302ff]. The following are all pseudo-partitives:

(5-58) a a number of giraffes
b a cup of cucumber
c a piece of gnu

whereas the following are partitive NPs:

(5-59) a many of the giraffes
b some of the cucumbers
c one of the gnus

Intuitively, we want to say that a partitive noun phrase describes an entity which is derived from (usually, is part of) some other entity; whereas a pseudo-partitive noun phrase describes a quantity of some substance. We will discuss partitives in section 5.5, since they involve the use of strategies for subsequent reference (they correspond to Prince's category of BRAND-NEW ANCHORED, where the anchor is a known entity).

In the present work, we analyse noun phrases like

(5-60) a ring of onion

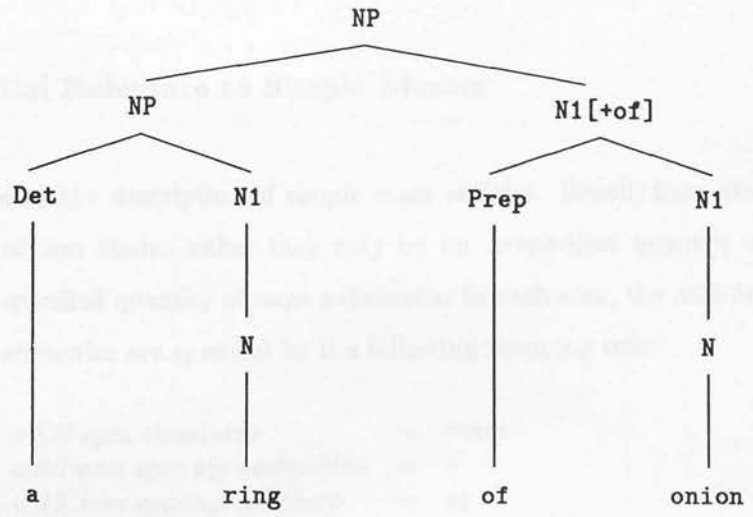


Figure 5.6: The Structure of Pseudo-Partitive Constructions

by means of the following rule:

(5-61) $NP \rightarrow NP\ N1[+of]$

This results in the structural analysis shown in figure 5.6.

The annotations required to distribute the semantic content through the structure are then as follows:

(5-62) $NP_1 \rightarrow NP_2\ N1[+of]$

$$\begin{aligned} <NP_2\ sem> &= <NP_1\ sem\ spec\ desc\ spec_1> \\ <N1\ sem> &= <NP_1\ sem\ spec\ desc\ spec_2> \\ <N1\ syn\ agr> &= <NP_1\ sem\ spec\ agr> \\ <NP_2\ sem\ status> &= <NP_1\ sem\ status> \end{aligned}$$

The alternate form

(5-63) an onion ring

is realized by means of a noun-compounding rule:

(5-64) $N1_1 \rightarrow N1_2\ N1_3$

$$\begin{aligned} <N1_2\ sem> &= <N1_1\ sem\ spec\ desc\ mod> \\ <N1_3\ sem> &= <N1_1\ sem\ spec\ desc\ head> \end{aligned}$$

To simplify things in the present work, we instead include a lexical entry for *onion* as an adjective, and analyse *an onion ring* syntactically as we would *a mouldy carrot*.

5.3.4 Initial Reference to Simple Masses

We now turn to the description of simple mass entities. Recall, from chapter 3, that masses can of two kinds: either they may be an unspecified quantity of some substance, or a specified quantity of some substance. In each case, the AGR features in the recoverable semantics are specified by the following mapping rule:

(5-65)
$$\begin{aligned} &\langle KB\ spec\ structure\rangle && = && mass \\ &\langle RS\ sem\ spec\ agr\ countable\rangle && = && - \\ &\langle RS\ sem\ spec\ agr\ number\rangle && = && sg \end{aligned}$$

However, the construction of the remainder of the RS structure is different in each of the two cases. First, we deal with masses of unspecified quantity.

Building the Recoverable Semantics

For simple masses, the KB structure corresponding to an ingredient described as

(5-66) some salt

would look like the following:

(5-67)
$$\text{KB} = \left[\begin{array}{l} index = x \\ state = s \\ spec = \left[\begin{array}{l} structure = mass \\ substance = salt \end{array}\right] \end{array}\right]$$

The algorithm for determining the content of initial references to simple masses is very simple:

If the STRUCTURE of an entity x is *mass* and no QUANTITY is specified:

- describe the SUBSTANCE of x; and
- describe any other locally-encoded properties of x.

We have a rule that picks up the substance:

(5-68) $<KB\ spec\ structure> = mass$
$<RS\ sem\ spec\ type\ category> = <KB\ spec\ substance>$

For our example, the resulting RS structure is as follows:

(5-69) RS =
$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = - \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = - \\ number = sg \end{bmatrix} \\ type = \begin{bmatrix} category = salt \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Building the Abstract Syntactic Structure

The same mapping rules as are used for simple individuals apply here, resulting in the following abstract syntactic structure:

(5-70) AS =
$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = - \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = - \\ number = sg \end{bmatrix} \\ desc = \begin{bmatrix} head = salt \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The basic NP rules described in section 5.2 then apply, resulting in the generation of the noun phrase *some salt*.

5.3.5 Initial Reference to Masses Specified by Quantity

Describing a mass whose quantity is specified is slightly more complicated. Suppose we have the knowledge base structure which might be described as

(5-71) two ounces of rice

The corresponding knowledge base structure is

$$(5\text{-}72)\quad \text{KB} = \begin{bmatrix} index = x \\ state = s \\ spec = \begin{bmatrix} structure = mass \\ substance = rice \\ quantity = \begin{bmatrix} unit = ounce \\ number = 2 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The required recoverable semantic structure is of the following general form:

$$(5\text{-}73)\quad \text{RS} = \begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \ldots \\ spec = \begin{bmatrix} agr = \ldots \\ quant = \begin{bmatrix} agr = \ldots \\ type = \ldots \end{bmatrix} \\ subst = \begin{bmatrix} agr = \ldots \\ type = \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The required KB→RS mapping rules are then as follows, where R is $<RS\ sem\ spec>$:

$$(5\text{-}74)\quad \begin{aligned} <R\ quant\ agr\ number> &= <KB\ spec\ quantity\ number> \\ <R\ quant\ agr\ countable> &= + \\ <R\ quant\ type\ category> &= <KB\ spec\ quantity\ unit> \\ <R\ subst\ agr\ number> &= sg \\ <R\ subst\ agr\ countable> &= - \\ <R\ subst\ type\ category> &= <KB\ spec\ substance> \end{aligned}$$

In the case of our example, the resulting recoverable semantic structure is:

(5-75)
$$\left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[\ given = - \ \right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = sg \\ countable = - \end{array}\right] \\ quant = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = 2 \\ countable = + \end{array}\right] \\ type = \left[\ category = ounce \ \right] \end{array}\right] \\ subst = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = sg \\ countable = - \end{array}\right] \\ type = \left[\ category = rice \ \right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

Building the Abstract Syntactic Structure

The relevant RS→AS mapping rules are as follows, where A is $<AS\ sem\ spec>$ and R is $<RS\ sem\ spec>$. First, the various agreement features are determined by the following rules:

(5-76)
$$\begin{array}{lcl} <A\ agr> & = & <R\ quant\ agr> \\ <A\ desc\ spec_1\ agr> & = & <A\ agr> \\ <A\ desc\ spec_2\ agr> & = & <R\ subst\ agr> \end{array}$$

Then, the basic algorithm described in section 5.2 for the embedding of properties is applied, first to build $<A\ desc\ spec_1\ desc>$ using the information specified in $<R\ quant\ type>$, and then to build $<A\ desc\ spec_2\ desc>$ using the information specified in $<R\ subst\ type>$.

This results in the following abstract syntactic structure:

(5-77)
$$\left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[\begin{array}{l} given = - \end{array}\right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} countable = + \\ number = 2 \end{array}\right] \\ desc = \left[\begin{array}{l} spec_1 = \left[\begin{array}{l} agr = \left[\begin{array}{l} countable = + \\ number = 2 \end{array}\right] \\ desc = \left[\begin{array}{l} head = ounce \end{array}\right] \end{array}\right] \\ spec_2 = \left[\begin{array}{l} agr = \left[\begin{array}{l} countable = - \end{array}\right] \\ desc = \left[\begin{array}{l} head = rice \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

Building the Surface Syntactic Structure

There are interesting differences in the linguistic behaviour of measure phrases such as our present example, and the pseudo-partitives we discussed earlier. First, compare the number agreements in the following:

(5-78) a one ring of onion is ...
b two rings of onion are ...

(5-79) a a house of stone is ...
b two houses of stone are ...

(5-80) a one ounce of cheese is ...
b four ounces of cheese is ...

In the case of pseudo-partitives, the embedded NP appears to control the number of the whole NP, but in measure phrases, this does not seem to be the case. Another difference between the two is exemplified by the following:

(5-81) a a ring of onion
b an onion ring

(5-82) a an ounce of cheese
b *a cheese ounce

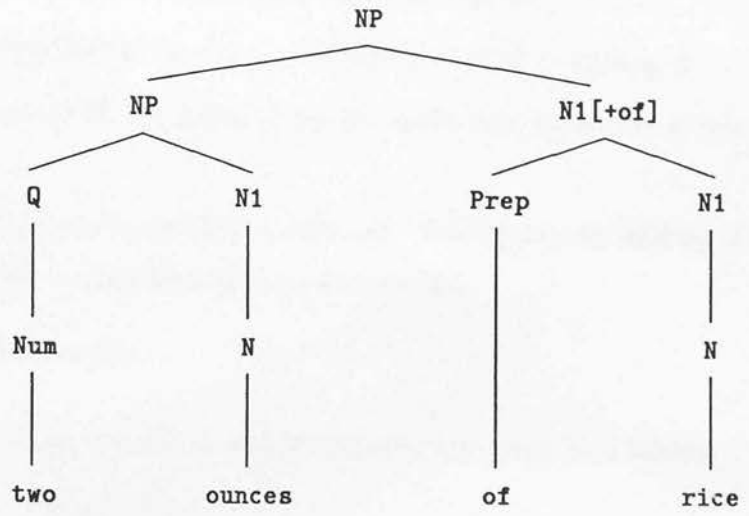


Figure 5.7: The Structure of Measure Phrases

These differences might lead us to postulate different surface syntactic structures for each. However, in the present work, we accord measure NPs the same analysis we presented for pseudo-partitives: we account for the difference in behaviour at the level of the semantic representation. This is as it should be, for the two kinds of expressions are *semantically* different: pseudo-partitives describe individuals, whereas measure phrases describe masses.

The surface syntactic structure we assign to *two ounces of rice* is then as shown in figure 5.7. The grammar rules required to construct this from the abstract syntactic structure are just those we presented above for pseudo-partitives (the rule which introduces the category Q will be introduced below).

5.3.6 Initial Reference to Sets of Unspecified Cardinality

We now turn to the process of making initial references to sets of entities. Recall from chapter 3 that there are various ways in which entities whose STRUCTURE is *set* may be represented:

- as sets of unspecified cardinality, as in *some carrots*;
- as sets of specified cardinality, as in *six aubergines*;
- as sets specified by quantity, as in *three pounds of courgettes*; or
- as sets specified by explicit listing of constituents, as in *200g of rice and peas*.

Different mapping rules are used in each case. We begin by considering sets of unspecified cardinality, as described by noun phrases like

(5-83) some carrots

The knowledge base structure corresponding to this entity is as follows:

(5-84)
$$\text{KB} = \left[\begin{array}{l} index = x \\ state = s \\ spec = \left[\begin{array}{l} structure = set \\ elements = \left[\begin{array}{l} structure = individual \\ substance = carrot \\ packaging = \left[\begin{array}{l} shape = carrot \\ size = regular \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

If a set does not have a specified cardinality, then the following rule is used to determine the AGR features:

(5-85)
$$\begin{array}{lcl} <KB\ spec\ structure> & = & set \\ <RS\ sem\ spec\ agr\ countable> & = & + \\ <RS\ sem\ spec\ agr\ number> & = & pl \end{array}$$

Much of the content of the corresponding recoverable semantic structure is determined by the ELEMENTS feature. The mapping rules used here are just like those used for other kinds of STRUCTURE, except that they make use of <*KB spec elements*> instead of <*KB spec*>: so, corresponding to the rule for simple individuals

(5-86)
$$\begin{array}{lcl} <KB\ spec\ substance> & = & <KB\ spec\ packaging\ shape> \\ <RS\ sem\ spec\ type\ category> & = & <KB\ spec\ substance> \\ <RS\ sem\ spec\ type\ props\ size> & = & <KB\ spec\ packaging\ size> \end{array}$$

we have the rule

(5-87) $\langle KB\ spec\ elements\ substance\rangle = \langle KB\ spec\ elements\ packaging\ shape\rangle$
$\langle RS\ sem\ spec\ type\ category\rangle = \langle KB\ spec\ elements\ substance\rangle$
$\langle RS\ sem\ spec\ type\ props\ size\rangle = \langle KB\ spec\ elements\ packaging\ size\rangle$

In the case of our *some carrots*, the resulting recoverable semantic structure is then:

(5-88) $$\text{RS} = \left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[\ given = -\ \right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} countable = + \\ number = pl \end{array}\right] \\ type = \left[\begin{array}{l} category = carrot \\ props = \left[\ size = regular\ \right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

The abstract syntactic and surface syntactic structures are then derived in the same way as for entities whose STRUCTURE is *individual*, so we will not repeat the details here.

5.3.7 Sets of Specified Cardinality

We next consider sets of specified cardinality, as in

(5-89) three carrots

Recall that the corresponding KB structure is as follows:

(5-90) $$\left[\begin{array}{l} index = x \\ state = s \\ spec = \left[\begin{array}{l} structure = set \\ cardinality = 3 \\ elements = \left[\begin{array}{l} structure = individual \\ substance = carrot \\ packaging = \left[\begin{array}{l} shape = carrot \\ size = regular \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

The rules used to construct an initial reference to sets of this kind are essentially the same as those presented for sets of unspecified cardinality, except for a number of minor

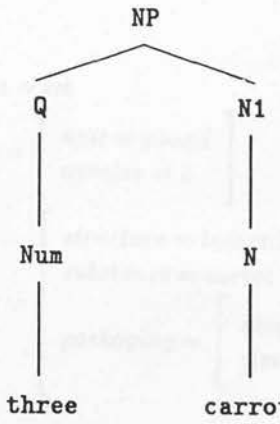


Figure 5.8: Sets specified by cardinality

differences. First, the KB→RS mapping rule which determines the AGR features includes the following:

(5-91) $<RS\ sem\ spec\ agr\ number> = <KB\ spec\ cardinality>$

The RS→AS rules are as before. To deal with explicit number, we have the following additional rules in the grammar:

(5-92)
$$\begin{array}{llll} NP \rightarrow Q\ N1 & <Q\ sem\ status> & = & <NP\ sem\ status> \\ & <NP\ syn\ agr> & = & <NP\ sem\ spec\ agr> \\ & <N1\ syn\ agr> & = & <NP\ syn\ agr> \\ & <Q\ syn\ agr> & = & <N1\ syn\ agr> \\ & <Q\ sem\ number> & = & <NP\ sem\ spec\ agr\ number> \\ & <N1\ sem> & = & <NP\ sem\ spec\ desc> \\ \\ Q \rightarrow Num & <Num\ sem> & = & <Q\ sem> \end{array}$$

Thus, we analyse the surface syntactic structure of *three carrots* as shown in figure 5.8.

5.3.8 Sets Specified by Quantity

Recall from chapter 3 that we can also specify sets in terms of a quantity, as in

(5-93) three pounds of carrots

The knowledge base structure corresponding to this entity is as follows.

(5-94)
$$\begin{bmatrix} index = x \\ state = s \\ spec = \begin{bmatrix} structure = set \\ quantity = \begin{bmatrix} unit = pound \\ number = 3 \end{bmatrix} \\ elements = \begin{bmatrix} structure = individual \\ substance = carrot \\ packaging = \begin{bmatrix} shape = carrot \\ size = regular \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Structures of this kind are dealt with in a manner essentially similar to that used for quantities of mass such as *two ounces of rice* and individuals such as *a ring of onion*; as ever, the differences are contained in the KB→RS rules. In this case, we use the following mapping rules (these are, in fact, a combination of rules that we have seen already). First, the QUANT feature in the recoverable semantics structure is determined as follows, where R is $<RS\ sem\ spec>$:

(5-95)
$$\begin{aligned} <R\ quant\ agr\ number> &= <KB\ spec\ quantity\ number> \\ <R\ quant\ agr\ countable> &= + \\ <R\ quant\ type\ category> &= <KB\ spec\ quantity\ unit> \end{aligned}$$

The SUBST feature is then built by the rules:

(5-96)
$$\begin{aligned} <R\ subst\ agr\ number> &= pl \\ <R\ subst\ agr\ countable> &= + \end{aligned}$$

along with the standard rules for describing individuals, applied to the construction of $<RS\ sem\ spec\ type\ subst\ type>$ instead of $<RS\ sem\ spec\ type>$. The resulting RS structure is as follows:

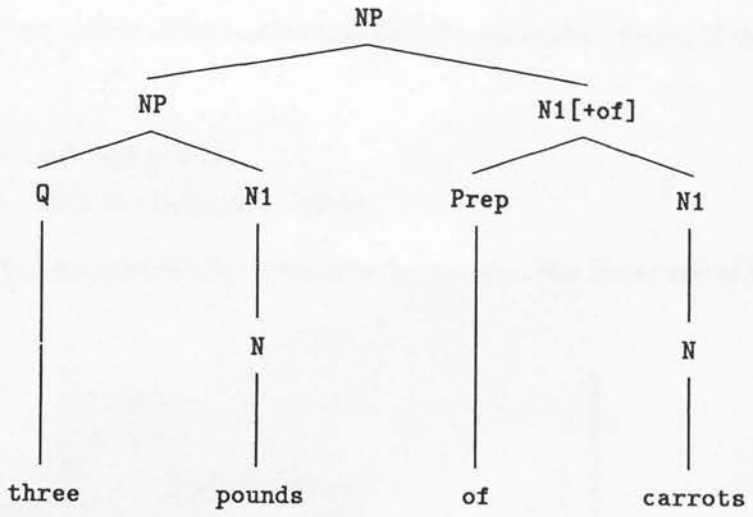


Figure 5.9: Sets specified by quantity

(5-97)

$$\left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[\begin{array}{l} given = - \end{array}\right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = pl \\ countable = + \end{array}\right] \\ quant = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = 3 \\ countable = + \end{array}\right] \\ type = \left[\begin{array}{l} category = pound \end{array}\right] \end{array}\right] \\ subst = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = pl \\ countable = + \end{array}\right] \\ type = \left[\begin{array}{l} category = carrot \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

The normal rules for constructing the abstract syntactic and surface syntactic structures apply, resulting in the surface syntactic structure shown in figure 5.9.

5.3.9 Sets Specified by Explicit Listing of Elements

Finally, we have to deal with those sets specified by an explicit listing of their elements, as in

(5-98) a salt and pepper
b 350g of raisins and sultanas

Recall that the knowledge base structures for entities like these are of the following form:

$$\text{(5-99)}\quad \text{KB} = \left[\begin{array}{l} index = x \\ state = s \\ spec = \left[\begin{array}{l} structure = set \\ quantity = \left[\begin{array}{l} unit = grams \\ number = 350 \end{array}\right] \\ constituents = [x_1, x_2] \end{array}\right] \end{array}\right]$$

where the constituent entities are described by distinct knowledge base structures:

$$\text{(5-100)}\quad \text{KB} = \left[\begin{array}{l} index = x_1 \\ state = s \\ spec = \left[\begin{array}{l} structure = set \\ elements = \left[\begin{array}{l} structure = individual \\ substance = raisin \\ packaging = \left[\begin{array}{l} shape = raisin \\ size = regular \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

$$\text{(5-101)}\quad \text{KB} = \left[\begin{array}{l} index = x_2 \\ state = s \\ spec = \left[\begin{array}{l} structure = set \\ elements = \left[\begin{array}{l} structure = individual \\ substance = sultana \\ packaging = \left[\begin{array}{l} shape = sultana \\ size = regular \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

The corresponding semantic structure includes specifications of the constituents of the set, derived in the normal way; these are collected together to provide a list as the value

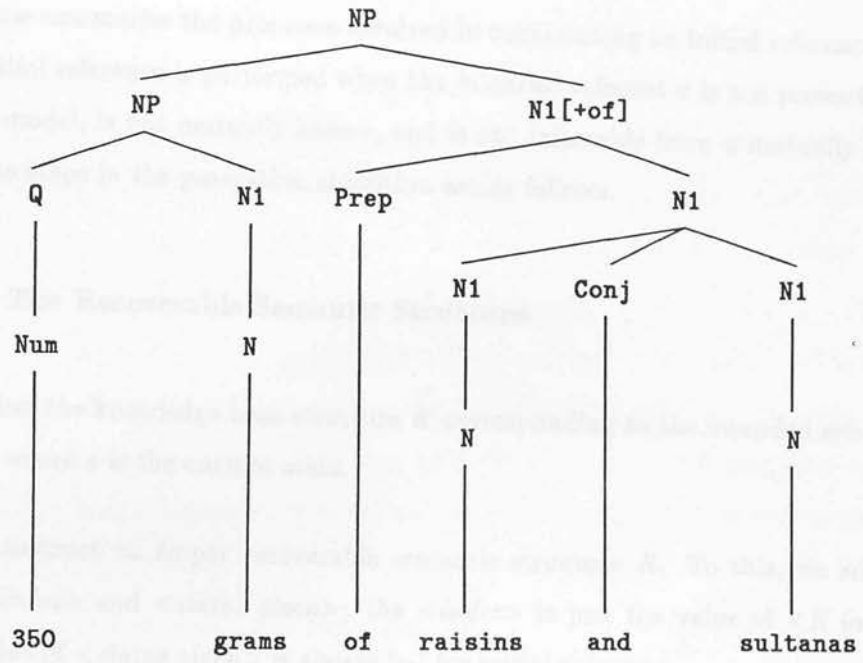


Figure 5.10: Conjoined N1s

of the SUBST feature in the recoverable semantic structure, with this list structure being maintained in the abstract syntactic structure. This is then realized by means of the following grammar rule:[10]

(5-102) $N1 \rightarrow N1_1, N1_2, \ldots, Conj\ N1_n$

$<N1_1\ sem> = <N1\ sem\ first>$

$<N1_2\ sem> = <N1\ sem\ second>$

...

$<N1_n\ sem> = <N1\ sem\ nth>$

where FIRST, SECOND and NTH are shorthand here for a mechanism that extracts the appropriate structure from a list of structures.

The resulting surface syntactic structure is then as in figure 5.10.

[10]The rule is not actually implemented as shown here: a different representation is used that unpacks the list recursively, using the basic list operations *first* and *rest*.

5.3.10 Generating Initial References

We can now summarise the processes involved in constructing an initial reference to an entity. Initial reference is performed when the intended referent x is not present in the discourse model, is not mutually known, and is not inferrable from a mutually known entity. The steps in the generation algorithm are as follows.

Building The Recoverable Semantic Structure

First, we find the knowledge base structure K corresponding to the intended referent x in state s, where s is the current state.

We then construct an empty recoverable semantic structure R. To this, we add the features <*index*> and <*status given*>: the <*index*> is just the value of <K *index*>, and the value of <*status given*> is always '–' for initial reference.

The semantic content of the recoverable semantic structure, i.e., the value of <R *sem*>, is then constructed. How this is done depends on the value of <K *spec structure*>, as follows.

First, we have to construct the appropriate values for <R *sem spec agr*> (i.e., the agreement features for the noun phrase that will result). The agreement features are determined by the following path equations, where the first equation in each case serves as a condition on the application of those following.

<*KB spec structure*>	=	*individual*
<*RS sem spec agr countable*>	=	+
<*RS sem spec agr number*>	=	*sg*
<*KB spec structure*>	=	*mass*
<*RS sem spec agr countable*>	=	–
<*RS sem spec agr number*>	=	*sg*
<*KB spec structure*>	=	*set*
<*RS sem spec agr countable*>	=	+
<*RS sem spec agr number*>	=	*pl*

Next, we can fill out the semantic content of the recoverable semantic structure using the information in K. Again, how this is done depends on the value of <K *structure*>:

- If K has structure *individual*, or it has structure *mass* and does not have a specified quantity, then we build up the contents of $<R$ *sem spec type*$>$ using the information in $<K$ *spec*$>$, as described in sections 5.3.3 and 5.3.4.
- If K has structure *mass* and has a value for $<$*spec quantity*$>$, then the value of $<R$ *sem spec quant*$>$ is determined using the information in $<K$ *spec quantity*$>$, and the value of $<R$ *sem spec subst*$>$ is determined using the other information in $<K$ *spec*$>$ as for unspecified quantities of mass, as described in section 5.3.5.
- If K has structure *set* and neither has any quantity specified, nor is specified in terms of particular constituents, then $<R$ *sem spec type*$>$ is constructed as for individuals and masses above, but using the information in $<K$ *spec element*$>$ instead of $<K$ *spec*$>$, as described in section 5.3.6. In addition, if $<K$ *spec cardinality*$>$ is specified, this information is added to $<R$ *sem spec agr*$>$, as described in section 5.3.7.
- If K has structure *set* and has a specified quantity, then $<R$ *sem spec quant*$>$ is constructed using the information in $<K$ *spec quantity*$>$, and $<R$ *sem spec subst*$>$ is constructed as using the information specified in $<K$ *spec element*$>$, as described in section 5.3.8.
- If K has structure *set* and has an explicit list of constituents, then $<R$ *sem spec subst*$>$ is constructed as a list of structures, where each is constructed in accordance with the current algorithm, as described in section 5.3.9. In addition, if K has a specified quantity, this is used to construct $<R$ *sem spec quant*$>$.

At the end of this process, we have a recoverable semantic structure that represents the semantic content of the noun phrase to be generated. Note that there are two basic types of RS structures: those which have a substructure addressed as $<$*sem spec type*$>$, and those which have two sister substructures addressed as $<$*sem spec quant subst*$>$. A third basic type, not covered here, has two sister substructures addressed as $<$*sem spec part*$>$ and $<$*sem spec set*$>$: we will consider these structures in section 5.5.

Building the Abstract Syntactic Structure

Given a recoverable semantic structure R, we construct an abstract syntactic structure A according to the following algorithm. Note that the same mapping rules are used to

build abstract syntactic structures: it is only at the level of constructing recoverable semantic structures that different mapping rules are required for initial and subsequent reference respectively.

First we construct an empty abstract syntactic structure *A*. To this, we copy across the structures <*index*>, <*sem status*>, and <*sem spec agr*> from *R*. We then construct the semantic content of the structure, i.e., the value of <*A sem*>, using the information in <*R sem spec*>. There are three parts to this algorithm, depending on the type of the recoverable semantic structure; two of these are described below, the procedures used for RS structures which have substructures <*sem spec part*> and <*sem spec set*> being described in section 5.5.

- If <*R sem spec type*> exists, then <*A sem spec desc*> is constructed from the information it contains: as described in section 5.2.3, this basically involves structuring the semantic elements in such a way that they match the structure of the syntactic construction that will ultimately be generated.
- If <*R sem spec quant*> exists, then the two structures addressed as <*A sem spec desc* $spec_1$> and <*A sem spec desc* $spec_2$> are constructed, the former using the information in <*R sem spec quant*>, and the latter using the information in <*R sem spec subst*>. In each case, the construction process is really a recursive application of this algorithm, as described in section 5.3.5.

Producing the Surface String

The complete noun phrase grammar can be found at the end of this chapter.

5.4 The Pronominalization Decision

> "Take a few eggs," the recipe goes on, "and carefully separate the whites from the yolks. Now whisk them into the mixture." The whites or the yolks? We compromise with a half of each.[11]

[11] From *The Original Michael Frayn*, Michael Frayn, The Salamander Press, Edinburgh, 1983, page 13.

The problem of pronominal reference resolution has always assumed a central importance in work on natural language interpretation. The corresponding problem in natural language generation is referred to as the PRONOMINALIZATION DECISION: that is, when is it appropriate to use a pronoun to refer to an entity?

As we noted at the beginning of this chapter, in order to adhere to the principle of adequacy, an anaphoric definite noun phrase must distinguish the intended referent from any other entities with which it might be confused. In order to achieve this goal, the noun phrase in question must provide sufficient semantic content to uniquely identify the intended referent. Pronouns are, apart from gender information, essentially free of semantic content; and so, intuitively, the use of the pronominal form suggests that the intended referent is so salient in the discourse that it is considered as an antecedent before any other entities are considered.

As we saw in chapter 2, syntactic constraints may provide some assistance with respect to the resolution of intra-sentential pronominal anaphora. However, for intersentential pronominal anaphora, things are much less clear cut. Many of the systems we discussed earlier make use of, or suggest making use of, Sidner's [1979] notion of focus in the generation of pronouns (see, for example, McDonald [1980:220]; Appelt [1982:129]; and McKeown [1982:127]), as reviewed in chapter 2. The approach taken in the present work is closer to that of Grosz, Joshi and Weinstein's [1983] notion of centering. Below, we draw a distinction between IMMEDIATE PRONOMINALIZATION and LONG DISTANCE PRONOMINALIZATION, and present the algorithm used in EPICURE for deciding on immediate pronominalization. We then suggest how long distance pronominalization might be explained in terms of the effects of discourse structure.

5.4.1 Immediate Pronominalization

Examination of English discourse shows that pronouns which refer to entities not mentioned in the current or previous sentence or utterance are extremely rare.[12] Noting this fact, we can classify occurrences of anaphoric third person pronouns in English as belonging to two categories, which we will call immediate and long distance pronominalizations respectively. We focus here on immediate pronominalization, and return to

[12]As we noted in chapter 2, in a large sample of pronouns considered by Hobbs, 98% of antecedents were found to be either in the same or previous sentence [Hobbs 1978:322–323].

consider long distance pronominalization later.

We define immediate pronominalization to be the use of a pronoun to refer to an entity to be found in the the IMMEDIATE CONTEXT of the discourse. Within EPICURE, the immediate context is represented by that part of the discourse model referred to as cache memory, as described in chapter 4. Because the computational model we have described only deals with single-clause sentences, this means we take the view that the immediate context consists of the preceding and current clauses of the discourse. As already suggested, this notion of immediate context is very close to what psycholinguists call syntactic memory. That pronominalization is, by and large, restricted to entities currently in syntactic memory is a view that has often been put forward in work on psycholinguistics. So, for example, Clark and Marshall [1981:44] claim that

> ...when the referents are recallable, or locatable, within immediate as opposed to long-term memory, the speaker can use a pronoun; otherwise, he cannot.
>
> [Clark and Marshall 1981:44]

Similarly, Johnson-Laird [1983:129] reports work by Wykes [1981] in which children were shown to assume that pronouns are co-referential with the subjects of previous clauses.

5.4.2 The Pronominalization Algorithm

Within EPICURE, the pronominalization decision is effectively distributed across two levels of processing. It is at the point when the recoverable semantic content is to be determined that the possibility of pronominalization is first considered. However, when the abstract syntactic structure is constructed, EPICURE may decide that a pronominalizable reference to an entity may be omitted altogether. This works as follows.

As we saw in chapter 4, we use the term CENTRE to refer to that entity which is the RESULT of the previously described operation. Thus, by the time the KB→RS rules begin construction of the recoverable semantics corresponding to the participants in an event to be described, one of these participants may already be marked as the centre: so, for example, in

(5-103) a Soak the beans.

b Drain <u>them</u>.

the recoverable semantic structure corresponding to the object of the second utterance before the semantic content for the participants has been decided will look like the following:[13]

(5-104) RS = $\left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[\begin{array}{l} given = + \\ centre = + \\ oblig = + \end{array}\right] \end{array}\right] \end{array}\right]$

The fact that the x is the CENTRE means that no semantic content is required in order to identify it; thus, a null TYPE feature is added:

(5-105) RS = $\left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[\begin{array}{l} given = + \\ centre = + \\ oblig = + \end{array}\right] \\ spec = \left[\begin{array}{l} type = \phi \end{array}\right] \end{array}\right] \end{array}\right]$

Nothing more is decided about producing a reference to this entity until the RS→AS mapping rules come into effect. Then, the following algorithm is applied:

- If a participant has the feature $[centre, +]$ then:
  - If that participant has the feature $[oblig, +]$, copy the recoverable semantic structure corresponding to the participant across to the abstract syntactic structure, and add AGR features consistent with the gender and number of x;
  - Otherwise, omit it from the abstract syntactic structure.

Thus, in the present example, the abstract syntactic structure corresponding to the pronominalized reference will be

[13] Recall from chapter 4 that the OBLIG feature is determined by the case frame of the particular verb to be used.

$$
(5\text{-}106)\quad \text{AS} = \left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[\begin{array}{l} given = + \\ centre = + \\ oblig = + \end{array}\right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} gender = neuter \\ number = pl \\ countable = + \end{array}\right] \\ desc = \phi \end{array}\right] \end{array}\right] \end{array}\right]
$$

The following grammar rule then produces the required surface string (where the case of the pronoun is determined by the sentence rule):

$$
\begin{array}{lllll} (5\text{-}107) & NP \rightarrow Pronoun & <Pronoun\ sem\ status> & = & <NP\ sem\ status> \\ & & <Pronoun\ sem\ spec> & = & <NP\ sem\ spec\ desc> \\ & & <Pronoun\ syn\ agr> & = & <NP\ sem\ spec\ agr> \end{array}
$$

The lexical entry for the pronominal form used here is as follows:

$$
(5\text{-}108)\quad \left[\begin{array}{l} phon = them \\ syn = \left[\begin{array}{l} cat = pronoun \\ agr = \left[\begin{array}{l} number = pl \\ gender = neuter \\ case = accusative \end{array}\right] \end{array}\right] \\ sem = \left[\begin{array}{l} status = \left[\begin{array}{l} given = + \\ unique = + \end{array}\right] \\ spec = \phi \end{array}\right] \end{array}\right]
$$

Now consider the case where the participant does not have the feature $[oblig, +]$, as determined by the verb case frame used in building the semantic structure. This results in the participant being omitted entirely from the abstract syntactic structure, resulting in utterances like (5-109b):

(5-109) a Fry the onions.
b Add the carrots ϕ.

where ϕ marks the omission of the participant, including the case marking preposition.

In addition, EPICURE will use pronominal reference to refer to an entity if that entity

was last mentioned in the previous sentence; thus, it is not only the centre that is pronominalizable (in other words, the algorithm used here is very close to the basic pronominalization algorithm proposed by Grosz, Joshi and Weinstein [1983]). This is still a very simplistic approach to pronominalization; however, in the current domain, the algorithm works satisfactorily.

A possible extension of the algorithm not considered here would be one that permits pronominal reference to events (cf. Webber [1988]). Since events are entities just as physobjs are, there is no reason why they should not serve as the referents of pronouns; the only problem is determining which events should be available as candidates for pronominal reference.

5.4.3 Long Distance Pronominalization

The algorithm above does not deal with instances of pronominalization where the antecedent is not in the immediate context; that is, it does not deal with long distance pronominalization as that term was defined earlier. In this section, we make some observations about long distance pronominalization, and suggest how it might be incorporated in the current framework.[14]

Grimes [1979] points out that some languages, such as Bacairi of Brazil, have two different third person pronominal forms. In such languages, one form is used to refer to what Grimes calls the LOCAL TOPIC, and the other form is used to refer to what he calls the GLOBAL TOPIC. This distinction seems to correspond to the two kinds of pronominalization we have suggested. We might suggest, then, that although English has only one pronominal form, this single surface form performs both functions. There then arises the question of how the hearer is to determine which function is being performed by a particular instance of the pronominal form. In what follows, we will suggest that uses of the pronominal form where it is not clear which function is being performed are INCONSIDERATE USES, and those where the function being performed is clear are CONSIDERATE uses (this terminology is adopted from Kantor [1977]). We make the claim that, if the pronoun is being used in a considerate fashion, the discourse

[14]Although compatible with the approach taken to pronominalization in the work described here, the solution proposed in this section is not implemented, since it requires a formal means of determining the global topic of a discourse—something which has so far escaped researchers in the area.

E: Good morning. I would like for you to re-assemble the compressor.

...

E: I suggest you begin by attaching the pump to the platform.

[...other subtasks]

E: Good. All that remains then is to attach the belt housing cover to the belt housing frame.

A: All right. I assume the hole in the housing cover opens to the pump pulley rather than to the motor pulley.

E: Yes, that is correct. The pump pulley also acts as a fan to cool the pump.

A: All right, the belt housing cover is on and tightened down.

[30 minutes and 60 utterances after beginning]

E: Fine. Now let's see if it works.

Figure 5.11: The Compressor Example (from Grosz [1977:23])

structure itself provides a means of determining which function is being performed.

In our discussion of discourse structure in chapter 2, we noted that full definite noun phrases are sometimes used in situations where the antecedent noun phrase is in the previous sentence. As an explanation for this phenomenon, we suggested that, if the hearer is presented with a pronoun to resolve, then this serves as an indication that the discourse segment containing the previous utterance has *not* been closed; thus, using a definite noun phrase is a way of indicating that a segment *has* closed. We also saw, however, a case which appeared to show use of a pronoun where a segment had just been closed, repeated here in figure 5.11. This may give a clue as to the situations in which long distance pronominalization can be used. However, since, as we noted, this particular example might be explained as deixis, we will first consider some other occurrences of long distance pronominalization before putting forward a hypothesis.

Examples of Long Distance Pronominalization

As noted above, examples of long distance pronominalization are rare. We present here two examples. One of these examples I find to involve an unacceptable use of

pronominal reference, and the other I find to be acceptable.[15] Unfortunately, intuitions on matters of this sort can be very difficult. We are not concerned here with relatively clear cut grammaticality judgements: the acceptability judgments required are much more subtle. In each case below, the pronominal references are all resolvable: the claim here, however, is that some are more easily resolved than others. That is, some instances of long distance pronominalization are considerate and some are inconsiderate; and the aim here is to define the circumstances required for considerate use.

In (5-110), we have a pronoun separated from its antecedent by two complete sentences:

(5-110) The first years of Henry's reign, as recorded by the admiring Hall, were given over to sport and gaiety, though there was little of the licentiousness which characterized the French Court. The athletic contests were serious but very popular. Masques, jousts and spectacles followed one another in endless pageantry. He brought to Greenwich a tremendously vital court life, a central importance in the country's affairs and, above all, a great naval connection.

[Halliday and Hasan 1976:14]

Here, Halliday and Hasan suggest, the pronoun *he* in the fourth sentence refers back to *Henry*, last mentioned in the first sentence.[16]

In the final sentence of the discourse shown in figure 5.12, the pronoun *it* is used to refer to *the aggressive nature of the child*, last mentioned at the beginning of this section of the discourse.[17]

The Effect of Discourse Structure on Pronominalization

The relative rarity of long distance pronominalizations might lead us to postulate one of two reasons for their occurrence. On the one hand, we might suggest that all instances

[15] I have informally questioned various individuals as to their judgements on the examples that follow, but I have not carried out a strictly controlled experiment: thus, it might be viewed as misleading if I were to talk of 'informant's judgements'. However, most people I have asked do share my judgements on the examples discussed, and so, although I will refer to the judgements as my own, it should be made clear that this does not imply that they are therefore idiosyncratic. I will address specific objections to the judgements as they arise in what follows.

[16] To be more precise, an entity—the first years of Henry's reign—to which Henry is explicitly associated is mentioned. We will ignore this complication, since the point remains if we replace the first sentence by a similar sentence which mentions Henry himself.

[17] At least, Reichman [1981:6] maintains that this is the antecedent of the pronoun. An alternative reading is that the antecedent is *acted that way* in the preceding clause.

R: Except, however, John and I just saw this two hour TV show.

M: Uh hum,

R: where they showed—it was an excellent French TV documentary—and they showed that, in fact, *the aggressive nature of the child* is not really that much influenced by his environment.

M: How did they show that?

R: They showed that by filming kids in kindergarten ...

M: Uh hum,

R: showing his behaviour among other children,

M: And then?

R: and showed him ten years later acting the same way, towards, um,

D: Well, of course, that's where he learns his behaviour, in kindergarten.

M: Oh, sure.

R: Now, another thing, it wasn't that he didn't have

J: What? What's that? What'd you say?

R: The aggressive child in kindergarten who acted the same way later on.

J: Yeah, he did.

R: Oh, and it was twins. The important thing was that there were two children from the same environment, whereas only one of the brothers acted that way. So you couldn't blame it on the child's home.

Figure 5.12: From Reichman [1981:5]

The first years of Henry's reign, as recorded by the admiring Hall, were given over to sport and gaiety, though there was little of the licentiousness which characterized the French Court.

The athletic contests were serious but very popular. Masques, jousts and spectacles followed one another in endless pageantry.

He brought to Greenwich a tremendously vital court life, a central importance in the country's affairs and, above all, a great naval connection.

Figure 5.13: A possible discourse structure for the 'Henry's reign' example

of long distance pronominalization are merely sloppy and inconsiderate language use: that in such instances, the speaker mistakenly assumes some entity to be the focus of attention for the hearer only because the speaker has been thinking about it, and thus uses a pronominal reference. Alternatively, we might suggest that long distance pronominalization is legitimate, but that it is rare because it may only occur under special circumstances.

As suggested earlier, pronoun use may indicate that the current discourse segment has not been closed. However, in both the compressor example and the nature/nurture example, there are what we might take to be explicit indications of discourse segment closure immediately before the use of the 'long distance' pronoun in each case: in the compressor example, we have the clue word *now* in *Now let's see if it works*, and in the nature/nurture example we have the clue word *so* in *So you couldn't blame it on the child*.

One of the claims made by Grosz and Sidner's [1987] theory of discourse structure is that it is not possible to use a pronoun to refer to an entity in a closed focus space. If this is correct, then, in conjunction with the assumption about the availability of antecedents for immediate pronominalization we adopted above, this means that the only place a pronoun *cannot* be used to refer to an entity in the preceding clause is in the first utterance immediately following the closure of a discourse segment.

We might hypothesise, then, that the only place it is legitimate to use long distance pronominalization—because it cannot be mistaken for immediate pronominalization—is in this location.

This claim could be generalised to also cover the deictic reading of Grosz's compressor example if we suggest that the relevant distinction is not between immediate and long-distance pronominalization, but between pronominal reference to entities in cache memory, and pronominal reference to entities which are in some sense globally salient, perhaps because of their relevance to the discourse, or perhaps because of their physical presence.

If the closing of the segment is clearly indicated by some other means, then the hearer will not look in cache memory for the antecedent, but instead will consider the most globally salient entities first. Of course, before this notion can be computationally

The first years of Henry's reign, as recorded by the admiring Hall, were given over to sport and gaiety, though there was little of the licentiousness which characterized the French Court.

For example, the athletic contests were serious but very popular; and masques, jousts and spectacles followed one another in endless pageantry.

All in all, he brought to Greenwich a tremendously vital court life, a central importance in the country's affairs and, above all, a great naval connection.

Figure 5.14: Clue words added to the 'Henry's reign' example

useful, we require a clear notion of what may be a global topic. However, it looks as though discourse structure may at least provide a way of explaining the few instances of long-distance pronominalization that do occur.[18]

What about Henry's Reign?

Without further argumentation, the above hypothesis does not explain the instance of long distance pronominalization in the 'Henry's reign' example. However, we might suggest that the text contains an embedded discourse segment, as shown in figure 5.13. If we then assume that the referent of *Henry* is the global topic of the discourse, then, under this analysis, the use of the pronoun *he* can be viewed as legitimate, since it is used to refer to the global topic in exactly those circumstances where this is permitted—except that the speaker has failed to provide any other clues as to the structure of the discourse. The plausibility of this explanation is increased if we add some 'clue words' to indicate the structure of the discourse, as shown in figure 5.14.

5.5 Subsequent Reference

We have now covered the mechanisms EPICURE uses in order to construct initial references to entities, and pronominal references to entities. In this section, we consider

[18] A computationally feasible alternative would be to suggest that, when a discourse segment is closed, cache memory is reloaded with whatever it contained immediately prior to that discourse segment being opened, much as a procedure call stack frame has any 'shadowing' of its variable names removed when control is returned to it. This would be taking the programming language metaphor too far, however, since there is no guarantee that the entities last mentioned in a discourse segment are necessarily the most salient or important entities in that discourse segment.

forms of subsequent reference other than pronominalization. In this category we include the following:

- simple definite noun phrase reference to individual, mass and set discourse entities, as in *the carrot*, *the salt*, and *the butterbeans*;
- subsequent references to sets by means of enumeration of the elements of the set, as in *the carrots, potatoes and broad beans*, and by means of superordinate terms, as in *the vegetables*;
- references to entities by means of their participation in events, as in *the eggs you boiled*; and
- references to entities which are introduced as being derived from other entities which have not been previously mentioned, as in *the kernels of a fresh ear of corn*.

We present the mechanisms required to generate each kind of reference given the appropriate knowledge base structures as input. In addition, we examine the processes required for each of the following kinds of reference:

- references to entities whose existence is inferrable from existing discourse entities, as in *the skin* when, for example, *a tomato* has just been mentioned;
- references to the complements of sets, as in *the rest of the vegetables*; and
- references to entities which are associated with, or part of, existing discourse entities, as in *one of the eggs*.

Limitations of the underlying knowledge representation used in the present system prevent us from being able to generate the latter kinds of referring expressions directly from knowledge base structures; however, we show the RS→AS rules required in each case. Not all of the above are subsequent references in the normal understanding of that term; however, each of these forms of reference involves processing that has something to do with subsequent reference, as we will see.

Before we go on to examine each kind of reference in detail, we first discuss the general mechanism by means of which EPICURE decides on the semantic content of a description. We view the generation of subsequent referring expressions as, in essence, distinguishing

an entity from that set of entities with which it might be confused. We call this set of potential distractors the CONTEXT. This approach then leads to two questions:

- How do we go about distinguishing an entity from a set of other entities?
- How do we determine the constituency of this set of other entities?

Below, we address both of these questions. First we consider how an entity can be distinguished from other entities, by introducing a notion of DISCRIMINATORY POWER. We then consider the question of how the relevant context is determined.

5.5.1 Telling Objects Apart

The principle of adequacy introduced at the beginning of this chapter requires that a subsequent reference contains sufficient information to distinguish an entity from all its potential distractors. At the same time, the principle of efficiency requires that we do not say more than we have to.

We describe here an abstract mechanism which, given an arbitrary set of entities, determines which properties of an entity are required in order to distinguish that entity from the set. We then describe the application of this mechanism in the generation of noun phrase referring expressions.

Kinds of Adjectives

Linguists and formal semanticists have long recognised that adjectives and the nouns they modify fit together in various ways. Different writers propose different categorisations, but most agree on the two following broad categories (although not always under the names used here):

- RELATIVE adjectives are those whose meaning depends upon the head noun which they modify: thus, for example, someone who might be described as *a small wrestler* is not necessarily describable as *a small jockey*. These adjectives are sometimes referred to as measure adjectives, since they perform a measuring function

with respect to some comparison class that is determined (sometimes indirectly) by the noun which the adjective modifies.

- ABSOLUTE adjectives, sometimes referred to as intersective adjectives or predicative adjectives, denote a set of objects in much the same way as head nouns do: thus, the denotation of a noun phrase containing a head noun and a single intersective adjective is the intersection of the set denoted by the adjective and the set denoted by the head noun.

It can be argued that the only absolute adjectives in English are DERIVED adjectives, like *four-legged*: all four-legged things have four legs. Adjectives like *red* are often taken to be absolute, but on closer inspection it is not clear if this is correct: the notion of *red* in the expression *red wine*, for example, appears to be quite different from that in *red telephone box*, and again from that in *red hair*.

There are, of course, many adjectives which fall into neither of the above categories. Some adjectives seem to negate the normal content of the noun they modify: for example, *fake* and *false* (an entity describable as *a fake gun* is not altogether a gun). There are also adjectives, like *alleged* and *possible* (as in *an alleged criminal* and *a possible solution*), which do not commit the speaker to asserting that the head noun with which they are used is true of entity referred to. Several authors have proposed formal treatments to distinguish different kinds of adjectives: see, for example, Kamp [1975], Bartsch [1975] and Keenan and Faltz [1978].

The mechanism described below applies, in the first instance, only to absolute adjectives, although we will use some relative adjectives as examples. In the current domain, this restriction is acceptable, since many of the adjectives used to describe ingredients are, in fact, derived adjectives, being derived from the verbs which describe processes that have been applied to those ingredients. The mechanism could perhaps be extended to cover relative adjectives by making use of the observation that many relative adjectives could be modelled using the notion of a perspective discussed at the end of chapter 3: such properties would then be relative to a perspective.

Suppose that we have a set of entities U such that

(5-111) $U = \{x_1, x_2, \ldots, x_n\}$

and that we wish to distinguish one of these entities, x_i, from all the others. Suppose, also, that the domain includes a number of attributes (a_1, a_2, and so on), and that each attribute has a number of permissible values (v_1, v_2, and so on); and that each entity is described by a set of attribute-value pairs. In a simple blocks world, for example, *basic category* and *colour* might be attributes; permissible values of these attributes are, respectively, *cube*, *pyramid* and *sphere*, and *red*, *green* and *blue*.

In order to distinguish x_i from the other entities in U, we need to find some set of attribute-value pairs which are together true of x_i, but of no other entity in U. This set of attribute-value pairs constitutes a DISTINGUISHING DESCRIPTION of x_i with respect to the context U. A MINIMAL DISTINGUISHING DESCRIPTION is then a set of such attribute-value pairs, where the cardinality of that set is such that there exists no other sets of attribute-value pairs of lesser cardinality which are sufficient to distinguish the intended referent.

We find a minimal distinguishing description by observing that different attribute-value pairs differ in the effectiveness with which they distinguish an entity from a set of entities. Suppose U has N elements, where $N > 1$. Then, any attribute-value pair true of the intended referent x_i will be true of n entities in this set, where $n \geq 1$. For any attribute-value pair $<a, v>$ that is true of the intended referent, we can compute the discriminatory power (notated here as F) of that attribute-value pair with respect to U as follows:

(5-112) $F(<a,v>, U) = \frac{N-n}{N-1}$ $\qquad 1 \leq n \leq N$

More generally, sometimes we want to distinguish some proper subset S of entities in U, where that subset contains more than one element (as in *the red blocks*). Discriminatory power is then calculated as follows:

(5-113) $F(<a,v>, U) = \frac{N-n}{N-m}$ $\qquad m \leq n \leq N$

where m is the cardinality of S.

F thus has as its range the interval [0,1], where a value of 1 for a given attribute-value pair indicates that the attribute-value pair singles out the intended referent from the context, and a value of 0 indicates that the attribute-value pair is of no assistance in singling out the intended referent.

As an example, suppose we have a context that contains five blocks, and two of these, including our intended referent, have the colour red: then,

(5-114) $F(<colour,\ red>, U) = 3/4$

Thus, the property of redness may help to distinguish our intended referent, but it does not do uniquely.

On the other hand, if all the blocks are red, then

(5-115) $F(<colour,\ red>, U) = 0$

Thus, in such a situation, the property of redness is of no value in distinguishing our intended referent.

If none of the other blocks are red, then

(5-116) $F(<colour,\ red>, U) = 1$

Thus, redness is the only property required in order to distinguish the intended referent.

Determining a Distinguishing Description Using Discriminatory Power

In order to produce a referring expression which adheres as much as possible to the principles of adequacy and efficiency, we have to make use of the properties with the greatest discriminatory power. Suppose we have a set P of attribute-value pairs which are true of the intended referent: the algorithm for selecting a minimal distinguishing description D, if one exists, for an entity x_i in a context U is then as follows. We accumulate the required attribute value pairs in D, which is initially an empty list, according to the the following procedure.

- If $P = \emptyset$, then a unique description cannot be constructed; return D.
- Otherwise, for each $<a_i, v_i> \in P$, calculate $F(<a_i, v_i>, U)$.
- Select the attribute-value pair with the highest F, and provided $F > 0$, add this attribute-value pair to D.
- If $F = 1$, then return D, which now contains a minimal distinguishing description;
- If $F > 0$, then let U be the set of $x_i \in U$ such that $<a_i, v_i>$ is true of x_i, and repeat the process with this new context.
- if $F = 0$, then no unique description can be constructed: return D.

The resulting description, if it exists, will satisfy the principles of efficiency and adequacy. Note that there may be more than one such description. This gives rise to the question of how we choose amongst different, equally minimal, distinguishing descriptions. There are a number of possible factors:

- the purpose of the description;
- there may be a conventional order in which properties are selected (this is borne out, at least to some extent, by the psychological evidence: see, for example, Herrmann and Laucht [1976]);
- properties may be selected according to some ordering which is context-dependent (this might include both global factors such as the particular domain of discourse and local factors such as properties previously used in the current discourse).

This problem does not arise in the current domain, and so the present work makes no particular claims in this respect.

Using Discriminatory Power in Constructing Noun Phrases

The abstract process described above requires some slight modifications before it can be used effectively for noun phrase generation. In particular, we should note that, in noun phrases, the head noun typically appears even in cases where it does not add any discriminatory power. For example, suppose there are six entities on a table, all of which are cups although only one is red: I am then likely to describe that particular cup as

as *the red cup* rather than simply *the red*. There are exceptions to this general rule, of course. For example, most days I buy my lunch at a local baker's, where I typically ask for *two brown cheese and tomatoes*: the sales assistant knows that I mean *two brown cheese and tomato rolls*.

The head noun in a noun phrase usually denotes the BASIC CATEGORY (after Rosch *et al* [1976]) or BASIC-LEVEL DESCRIPTOR of an entity: this is the 'level of abstraction at which the organism can obtain the most information with the least cognitive effort'. The algorithm described above for calculating discriminatory power assumes that there is a sense in which all properties are equal: that is, it is only the relative discriminatory power, and nothing else, that causes a particular property to be selected. However, in real language use, there are other factors that play a role in the choice of which properties should be used in describing an entity. That some properties are more useful than other properties which have the same discriminatory power is certainly true where *epistemic* identification is required: suppose there are three cups on the table, and it is mutually known that only one is red, and that the red cup is the only one that has a green sticker on its base. The discriminatory power of these two properties (being red, and having a green sticker on the base) is the same, but—if you have to actually identify the cup in question—the property of being red is, in most circumstances, more useful than the property of having a green sticker on its base.

Thus, in order to implement the above algorithm, we always first add to D (the description being constructed) that property of the entity denoted by its CATEGORY feature. In many cases, this means that no further properties need be added.

Using Discourse Structure to Reduce the Context

We are not yet in the clear. Although we now have a mechanism which, given a context of entities U, permits us to determine how best to distinguish an entity in this context from the other elements of the context, we have not yet said anything about how the context set itself is determined.

The simplest solution would be to take the entire contents of the discourse model—i.e., every entity that has been mentioned in the discourse—to be the relevant context.

However, the larger the context with respect to which an entity must be distinguished, the more computationally expensive the process of distinguishing the intended referent will be, and so it is sensible to look for some way of cutting down the size of the context. An intuitively appealing solution would be to order the entities that have been mentioned in the previous discourse in terms of salience: this was the essence of Lewis's suggestion discussed in chapter 2. However, as we pointed out there, it is not at all clear what metric of saliency should be used.

One candidate sometimes suggested in the literature is *recency of mention*: in constructing a referring expression, the intended referent need only be distinguished from those entities which have been mentioned since the intended referent was last mentioned. Given that our discourse model is already partitioned into separate focus spaces, we might suggest that entities last mentioned in focus spaces near the top of the focus stack are more salient than those last mentioned in focus spaces further down the focus stack. That is, taking FS_n to be the set of entities introduced in focus space n (where n is the ordinal position of the focus space on the stack, with the 0th space being the space on the bottom of the stack), and taking s to be a function that determines the salience of a set of entities; then, if FS_T is the focus space on the top of the focus stack, the following holds:

(5-117) $s(FS_T) > s(FS_{T-1}) > \ldots > s(FS_0)$

In fact, this is a stronger claim than the evidence of Grosz and Sidner's examples (discussed in chapter 2) supports. As it stands, the evidence supports only the claim that

(5-118) $s(FS_T) > s(FS_{T-1})$

Thus, an alternative generalisation consistent with the data would be

(5-119) $[s(FS_T) > s(FS_{T-1})] \wedge [s(FS_T) > s(FS_{T-2})] \wedge \ldots [s(FS_T) > s(FS_0)]$

i.e., only the topmost focus space is more salient than any other, and nothing is known of the relative salience of the others.

Determining the correct answer to this question (if, indeed, discourse structure has any real effect on this at all) is beyond the scope of the present work. In the present domain,

since the number of entities we are dealing with is relatively small, it is adequate to take the global working set to be the context.

5.5.2 Subsequent Reference to Individuals

We are now in a position to describe the processes EPICURE uses in constructing subsequent references to entities. We begin by considering the simpler forms of subsequent reference.

Suppose we wish to generate a subsequent reference to an entity x, which has already been mentioned in the discourse. The discourse model will include a knowledge base structure which specifies all the information that has been provided about that entity in the discourse so far. So, for example, we might have

(5-120) KB =
$$\left[\begin{array}{l} index = x \\ spec = \left[\begin{array}{l} structure = individual \\ substance = carrot \\ packaging = \left[\begin{array}{l} shape = carrot \\ size = regular \end{array}\right] \\ mouldy = + \end{array}\right] \end{array}\right]$$

as the representation within the discourse model for an entity that was introduced as *a mouldy carrot*.

In order to make a subsequent reference to this entity, we have to construct a minimal distinguishing description, as described in the previous section. The basic algorithm is as follows:

To refer to an entity x where that entity is already present in the discourse model:

- Build the basic RS structure with the appropriate AGR features and a value of + for GIVEN.
- Locate the context U within which x is to be distinguished.
- Determine the set of properties L to be used to identify x, and add these to the RS structure.

- If L provides a distinguishing description, then set UNIQUE to have a value of +; otherwise, set UNIQUE to have a value of −.

Recall that we first consider the property denoted by the head noun (in this case, the SUBSTANCE, since this is the same as the SHAPE of the PACKAGING) for its discriminatory power. If this is the only carrot that has been mentioned in the discourse, then

(5-121) $F(<substance,\ carrot>, U) = 1$

and so the only information that needs to be specified in the recoverable semantic structure is the SUBSTANCE. The resulting recoverable semantic structure is then:

(5-122) $$\text{RS} = \begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = + \\ unique = + \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = sg \end{bmatrix} \\ type = \begin{bmatrix} category = carrot \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The mapping to abstract syntactic structure and the subsequent realization by means of the grammar are as before.

In the case where there is another carrot in the context, then

(5-123) $F(<substance,\ carrot>, U) < 1$

(the exact value of F depends on the number of entities in the context). If this other carrot is not mouldy, then adding the property *mouldy* to the recoverable semantic structure will result in a distinguishing description.

However, if both carrots are mouldy, and there are no other properties that can be used to distinguish the intended referent, then we cannot use a definite referring expression to pick out the intended referent. Suppose there are no other carrots in the context than the two mouldy ones: then the *mouldy* property will not have been added to the set of properties to be realized, since it does nothing to cut down the set of possible

referents. The recoverable semantic structure is then as follows:

(5-124) RS =
$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = + \\ unique = - \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = sg \end{bmatrix} \\ type = \begin{bmatrix} category = carrot \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Note that if there is no way to distinguish one of these carrots from the other, then it does not matter which of the two the hearer believes the intended referent to be. If it did matter, then this would be on the basis of some property that one carrot possessed while the other did not; and if this was the case, then that property could be used in the description.[19] Thus, the intended referent can be safely described as either of the following:

(5-125) a a carrot
b one of the carrots

The algorithm used is then as follows:

- Try to build a unique description.
- If this is not possible, then either:
  - Describe the entity as non-unique; or
  - Describe the entity as one of the set picked out by the description.

In the present system, we generate the first of these. The recoverable semantic structure is then:

[19] More precisely, the referring expression is non-specific, in the sense discussed in chapter 2: the speaker is unlikely to have a particular carrot in mind in a situation like this. However, since we do not provide mechanisms for dealing with non-specific reference, we assume that the speaker does have a specific carrot in mind.

(5-126) RS =
$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = - \\ unique = - \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = sg \end{bmatrix} \\ type = \begin{bmatrix} category = carrot \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The resulting noun phrase generated is

(5-127) a carrot

Currently, EPICURE will not generate the second alternative, since this requires the intended referent and the set of which it is part to both be available as elements of the working set: this is not possible, given the current restriction on the constituency of the working set that all the elements are disjoint. This means that we are unable to specify the appropriate KB→RS rules; however, we can specify the rest of the processing that would be used. The recoverable semantic structure is then as follows:

(5-128)
$$\begin{bmatrix} index = x_2 \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = + \\ unique = - \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = sg \end{bmatrix} \\ type = \begin{bmatrix} part = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = sg \end{bmatrix} \\ type = \begin{bmatrix} category = carrot \end{bmatrix} \end{bmatrix} \\ set = \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where the recoverable semantic structure of the embedded set is as follows:

(5-129) RS =

$$\begin{bmatrix} index = x_1 \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = + \\ unique = + \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = pl \end{bmatrix} \\ type = \begin{bmatrix} category = carrot \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The STATUS and AGR features in the surface semantic structure are then determined in the normal way:

(5-130)
$$\begin{aligned} &<AS\ index> &&= <RS\ index> \\ &<AS\ sem\ status> &&= <RS\ sem\ status> \\ &<AS\ sem\ spec\ agr> &&= <RS\ sem\ spec\ agr> \\ &<AS\ sem\ spec\ desc\ spec\ agr> &&= <AS\ sem\ spec\ agr> \end{aligned}$$

The recoverable semantic structure above can be glossed as describing *a carrot of the carrots*. To remove the obvious redundancy here, the following rule applies:

(5-131)
$$\begin{aligned} &<RS\ sem\ spec\ part\ type> &&= <RS\ sem\ spec\ set\ sem\ spec\ type> \\ &<AS\ sem\ spec\ desc\ spec\ desc\ head> &&= \phi \end{aligned}$$

The structure $<AS\ sem\ spec\ desc\ set>$ is constructed from $<RS\ sem\ spec\ set>$ in the normal way. The resulting abstract syntactic structure is as follows:

(5-132)

$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = - \\ unique = - \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} number = sg \\ countable = + \end{bmatrix} \\ desc = \begin{bmatrix} spec = \begin{bmatrix} agr = \begin{bmatrix} number = sg \\ countable = + \end{bmatrix} \end{bmatrix} \\ desc = \begin{bmatrix} head = \phi \end{bmatrix} \\ set = \begin{bmatrix} index = x_1 \\ sem = \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

with the embedded structure being as follows:

(5-133) AS =
$$\begin{bmatrix} status = \begin{bmatrix} given = + \\ unique = + \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} number = pl \\ countable = + \end{bmatrix} \\ desc = \begin{bmatrix} head = carrot \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

This structure is then unified with the following grammar rule:

(5-134) $NP_1 \rightarrow NP_2\ PP[+of]$

$<NP_2\ sem\ status> = <NP_1\ sem\ status>$

$<NP_2\ sem> = <NP_1\ sem\ spec\ desc\ spec>$

$<PP\ sem> = <NP_1\ sem\ spec\ desc\ set>$

The word *one* is defined in the dictionary as an indefinite pronoun, as follows:

(5-135)
$$\begin{bmatrix} phon = one \\ syn = \begin{bmatrix} cat = pronoun \\ agr = \begin{bmatrix} number = sg \end{bmatrix} \end{bmatrix} \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = + \\ unique = - \end{bmatrix} \\ spec = \begin{bmatrix} desc = [] \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The resulting surface syntactic structure is then as in figure 5.15.

Subsequent references to masses and sets are performed in essentially the same fashion. References to sets can specify the cardinality explicitly, as in

(5-136) the three carrots

This noun phrase is assigned the surface syntactic structure in figure 5.16.

5.5.3 Referring to Sets and Complements

Sometimes EPICURE needs to make a subsequent reference to a set of entities, where the constituents of that set are made of different basic substances. In such cases, a

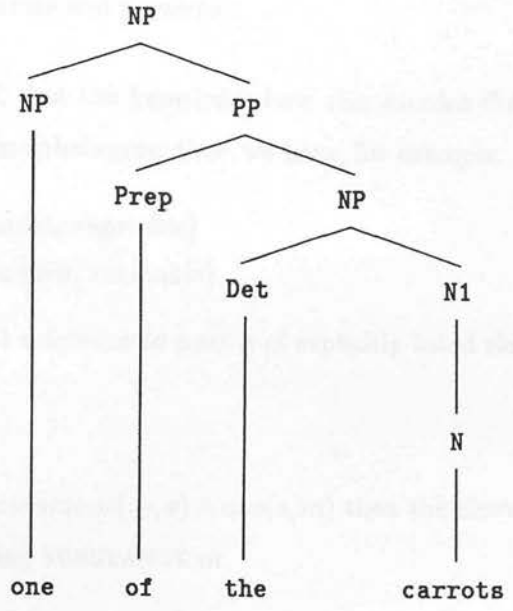


Figure 5.15: Non-unique Descriptions

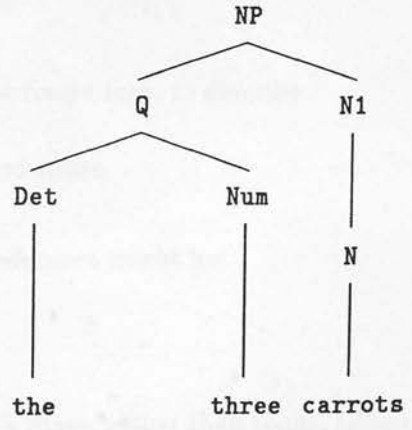


Figure 5.16: Definite sets specified by Cardinality

superordinate term can often be used. Suppose, for example, the following ingredient has been introduced into the discourse:

(5-137) 2lbs of carrots and parsnips

Recall, from chapter 3, that the knowledge base also encodes the hierarchical relationships that hold between substances; thus, we have, for example:

(5-138) a ako(carrot, vegetable)
b ako(parsnip, vegetable)

Whenever a subsequent reference to a set x of explicitly listed elements is required, the following holds:

- if $\exists m \exists s \forall x_i \in x \; substance(x_i, s) \wedge ako(s, m)$ then the elements of the set can be described as having SUBSTANCE m.

Before this property can be used to describe the set, we have to ensure that there are no other entities in the relevant context which can be described by the superordinate; thus, the superordinate substance is treated just like any other property by the discriminatory power mechanism.

In the present example, the resulting subsequent reference would then be

(5-139) the vegetables

If we were to use a superordinate term to describe

(5-140) 2lbs apples and pears

the resulting subsequent reference might be:

(5-141) the fruit

Note that this expression is mass, rather than count, although it really describes a set. The current mechanisms cannot deal with this; however, since what we have here is basically a mismatch between semantic and syntactic countability, this would be dealt with by the RS→AS mapping rules, in the same way as disagreements between semantic and syntactic number would be handled, as mentioned in section 5.2.3.

The mechanisms here could be extended to deal with the generation of referring expressions like

(5-142) the remainder of the vegetables

by means of the following algorithm, where x is the intended referent (a set of vegetables in the current example):[20]

- Find a superordinate term to describe the elements of x;
- If there are other ingredients in the recipe which can also be described by this superordinate term, but each such ingredient has already been used in the recipe, then the expression *the remainder of* ... can be used to describe x.

This could be implemented by giving each ingredient a USED attribute, whose initial value is '–'. The basic discriminatory power algorithm would then identify x in the example above as *the unused vegetables*.

5.5.4 References to Containing Inferrables

Some entities are referred to in terms of their associations with other entities, as in

(5-143) a a half of the cucumber
b the skin of the avocado

Note that the entity with which the intended referent is associated need not be already present in the discourse model: thus, we find ingredients specified at the start of a recipe in the same way, except that an indefinite noun phrase, rather than a definite noun phrase, is used to refer to the 'containing' entity:

(5-144) a the juice of a lemon
b the kernels of a fresh ear of corn

We handle both kinds of references in essentially the same way. The basic algorithm is as follows.

To refer to an entity which has an ancestor, where the entity itself has not been men-

[20] The basic approach here is similar to that used in Davey's program to construct noun phrases of the form *the other* ..., as described in chapter 2.

tioned before:

- If the ancestor has already been mentioned, build a recoverable semantic structure as for any other already mentioned entity.
- If the ancestor is new to the discourse, build a recoverable semantic structure as for initial reference.
- Describe the intended referent as being part of the ancestor.
- Set the intended referent's <*status given*> attribute to '+'.

Suppose, for example, we have an ingredient which is to be introduced as *the juice of a lemon*. This will be represented by the following KB structure:

$$\text{(5-145)} \quad \text{KB} = \begin{bmatrix} index = x \\ spec = \begin{bmatrix} structure = mass \\ part = juice \\ ancestor = x_1 \end{bmatrix} \end{bmatrix}$$

where the ancestor x_1 is represented by the following KB structure:

$$\text{(5-146)} \quad \text{KB} = \begin{bmatrix} index = x \\ spec = \begin{bmatrix} structure = individual \\ substance = lemon \\ packaging = \begin{bmatrix} shape = lemon \\ size = regular \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

The algorithm used is then as follows:

- If the intended referent has a SUBSTANCE, then describe the entity in terms of this.
- Otherwise, build a RS structure where <*RS sem spec part*> is derived from the KB that represents the intended referent, and <*RS sem spec set*> is derived from the KB that represents the ancestor.

The resulting recoverable semantic structure is then as follows:

(5-147)
$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = + \\ unique = + \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} number = sg \\ countable = - \end{bmatrix} \\ part = \begin{bmatrix} agr = \begin{bmatrix} number = sg \\ countable = - \end{bmatrix} \\ type = \begin{bmatrix} category = juice \end{bmatrix} \end{bmatrix} \\ set = \begin{bmatrix} index = x_1 \\ sem = \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where the embedded structure is as follows:

(5-148)
$$\text{RS} = \begin{bmatrix} status = \begin{bmatrix} given = - \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} number = sg \\ countable = + \end{bmatrix} \\ type = \begin{bmatrix} category = lemon \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Entities introduced by reference to their ancestors are always assumed given. The same basic mechanisms are also used when the ancestor itself has already been mentioned in the discourse; thus, we treat examples like

(5-149) the skin of the avocado

in the same way.

The RS→AS rules and grammar rules used here have already been described. The resulting noun phrases are analysed syntactically as in figure 5.17.

Sets specified as subsets of other sets are analysed syntactically in the same way: thus,

(5-150) four ounces of the lentils

has the structure shown in figure 5.18.

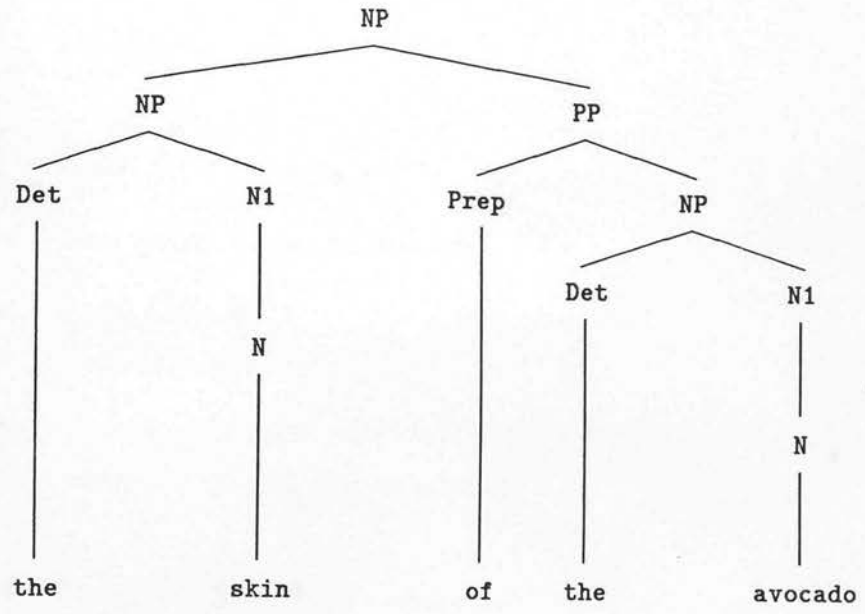


Figure 5.17: The Structure of Partitives

5.5.5 Referring to Inferrable Entities

In the previous section, we saw that entities introduced by reference to their ancestors, as in

(5-151) a the juice of a lemon
b the skin of the avocado

are assumed given. In this section, we consider references to INFERRABLE ENTITIES, i.e., entities whose existence can be inferred from the known existence of other entities, in such a way that not only are they assumed given, but there is also no requirement to mention the ancestor.[21] So, for example, we find things like:

(5-152) a Boil the egg.
b Remove the shell.

Examples of this kind appear, at first glance, relatively easy to explain: eggs typically

[21]The material described here is currently being implemented, and does not figure in the version of EPICURE described in chapter 6.

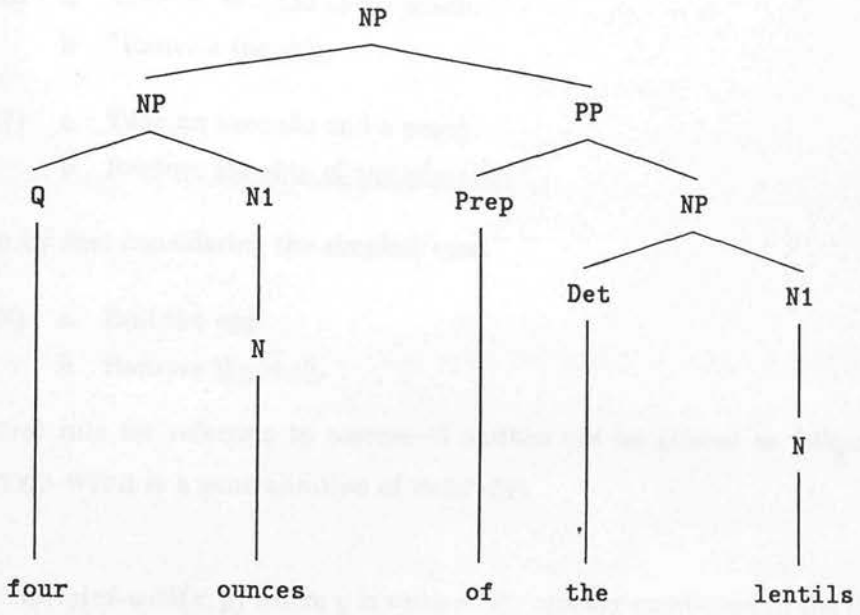


Figure 5.18: Sets derived from other Sets

have shells. Similarly, every orange has a number of segments, permitting the use of the referring expression in (5-153b):

(5-153) a Take an orange.
b Coat one of the segments with sugar.

We might suppose, then, that the generation of expressions like these simply requires us to add a rule along the following lines:

(5-154) To refer to x_1, where x_1 is a part of x_2, if the hearer can be assumed to infer that the ancestor of x_1 is x_2, then it is sufficient to describe x_1 simply by its part name.

Notice that any implementation of this rule has to take account of the following data. First, if the ancestor has more than one part of the kind in question, then a definite determiner is not appropriate:

(5-155) a Take an orange.
b *Coat the segment with sugar.

Second, if there is more than one possible ancestor, a more detailed description may be

required:

(5-156) a Take an avocado and a peach.
b *Remove the skin.

(5-157) a Take an avocado and a peach.
b Remove the skin of the avocado.

We begin by first considering the simplest case:

(5-158) a Boil the egg.
b Remove the shell.

Our general rule for referring to associated entities can be glossed as follows, where ASSOCIATED-WITH is a generalisation of PART-OF:

- if *associated-with*(x, y) where y is some entity already mentioned in the discourse, then this is a property that can be used in the description.

At its simplest, this would give us the following information:

(5-159) shell$(x) \wedge$ associated-with$(x, y) \wedge$ egg(y)

which, if realized, would result in any of the following:

(5-160) a the shell of the egg
b the egg shell
c the egg's shell

However, the hearer can be assumed to have the following general knowledge:

(5-161) $\forall x$ egg$(x) \supset [\exists y$ shell$(y) \wedge$ associated-with$(y, x)]$

The system must first then ask: can the hearer infer the existence of the intended referent? Given that the hearer has the relevant general knowledge, and the fact that the egg in question is already in the discourse model, we add associated entities to the discourse model. Thus, we add the information that:

(5-162) $\exists x$ shell$(x) \wedge$ associated-with(x, y)

In order to satisfy the principle of efficiency, EPICURE tries to obey the following rule:

- If the hearer already knows of the existence of an entity, do not bother to describe any properties of it except those required to identify it.

As we saw earlier in this section, we generally identify an entity by means of its basic category, plus any properties required to distinguish it from other entities it might be confused with. So, in the present case, all we have to say is

(5-163) the shell

The basic point is this: if we want to refer to an associate of an entity x that is already present in the discourse model, we add all the associates of x whose existence the hearer is able to infer to the discourse model (but not to cache memory, since this might license pronominalization). Once this has been done, the standard subsequent reference algorithms described earlier apply. In the present example, it is as if the discourse had included (5-164a):

(5-164) a There is a shell.
b Boil the egg.
c Remove the shell.

In the case of the following example

(5-165) a Take an orange.
b Coat one of the segments with sugar.

we might have the following general rule:

(5-166) $\forall x$ orange$(x) \supset$
$[\exists y$ set$(y) \wedge [\forall z\ z \in y \supset$ segment$(y)] \wedge$ associated-with$(y, x)]$

That is, every orange has a set of segments. Since EPICURE wants to refer to a particular segment here, we have the following information:

(5-167) segment$(x) \wedge x \in z \wedge$ set$(z) \wedge$ associated-with$(z, y) \wedge$ orange(y)

Thus, we must also search for associations other than direct ones:

(5-168) $\forall x, z\ x \in z \wedge \exists y$ associated-with$(z, y) \supset$
associated-with(x, y)

In order to generate the referring expression in question, we add to the discourse model an entity representing the set of segments:

(5-169) $\exists z\ \text{set}(z) \wedge \forall y\ y \in z \supset \text{segment}(y)$

Thus, the set of segments can be assumed known to the hearer, but no single element of this set is known uniquely. Since

(5-170) $\text{segment}(x) \wedge x \in z$

we can make use of a reference strategy which suggests, wherever possible, referring to newly-introduced entities as members of already-mentioned sets. This would result in the generation of the noun phrase

(5-171) one of the segments

Finally, consider:

(5-172) a Take an avocado and a peach.
b Remove the skin of the avocado.

This example shows that, if we add any associates to the discourse model, we also have to add the associates of other equally salient entities. The referring expression construction algorithms then have to take these other associates into account.

5.5.6 Referring to Participation in Events

The fact that an entity x has participated in an event e is just another property that is available for incorporation in a description.[22]

To enable this information to be used, each entity in the knowledge base is augmented with a PARTICIPATES-IN attribute, which lists the events in which the entity has participated. The property of having participated in a specific event can then be used to describe that entity. So, for example, if a particular carrot has participated in a number of events, we would have the following knowledge base structure:

[22] Again, the mechanisms described in this section do not figure in the version of EPICURE described in chapter 6.

$$(5\text{-}173)\quad \text{KB} = \begin{bmatrix} index = x \\ spec = \begin{bmatrix} structure = individual \\ substance = carrot \\ packaging = \begin{bmatrix} shape = carrot \\ size = regular \end{bmatrix} \\ participates\text{-}in = [e_3, e_2, e_1] \end{bmatrix} \end{bmatrix}$$

Suppose the event e_3 was described as

(5-174) Grate the carrot.

Then, the carrot x might be described in the following ways:

(5-175) a the carrot you grated
b the carrot that was grated

Thus, the event in which it participated can be described with a number of different degrees of specificity.

From this, we might build the following recoverable semantic structure:

$$(5\text{-}176)\quad \begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = + \\ unique = + \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = pl \end{bmatrix} \\ type = \begin{bmatrix} category = carrot \\ props = \begin{bmatrix} participated - in = \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

where the embedded PARTICIPATED-IN structure is as follows:

$$(5\text{-}177)\quad \text{RS} = \begin{bmatrix} index = e \\ sem = \begin{bmatrix} substance = grating \\ participants = \begin{bmatrix} agent = hearer \\ object = x \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

When the abstract syntactic structure is constructed, the embedded object here would be elided, with the resulting structure being as shown in figure 5.19. The full detail of

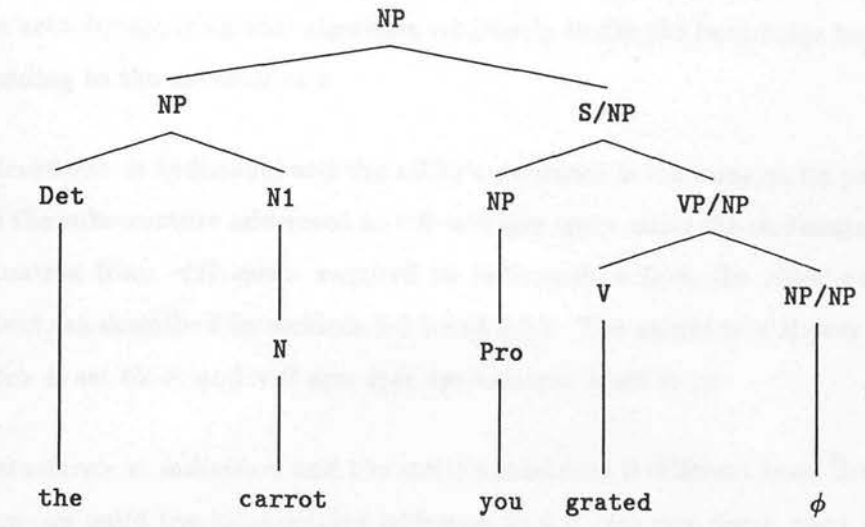


Figure 5.19: Relative Clauses

the mechanisms required to implement this have yet to be worked out, but the basic idea should be clear.

5.5.7 Generating Subsequent Reference

We summarise here the procedures used for the generation of subsequent reference, not including pronominal reference. Only the process of building a recoverable semantic structure is described below: the construction of the corresponding abstract syntactic structures is as described at the end of section 5.3.

Given an input knowledge base structure K corresponding to the intended referent x, the processes used to build a recoverable semantic structure for subsequent reference are as follows.

First, we build a recoverable semantic structure whose <*index*> is the index of the intended referent, and whose <*sem status given*> attribute has the value '+'.

What we do next depends on the structure of the intended referent.

If x has an ancestor, we build the substructure addressed as <*R sem spec part*> using the information contained in <*K spec*>, and we build the substructure addressed as <*R sem spec set*> by applying this algorithm recursively to the the knowledge base entity corresponding to the ancestor of x.

If <*K structure*> is *individual* and the entity's substance is the same as its packaging, we build the substructure addressed as <*R sem spec type*> using the minimum amount of information from <*K spec*> required to distinguish x from the other entities in the context, as described in sections 5.5.1 and 5.5.2. The attribute <*R sem spec agr countable*> is set to +, and <*R sem spec agr number*> is set to *sg*.

If <*K structure*> is *individual* and the entity's substance is different from the entity's packaging, we build the substructure addressed as <*R sem spec type*> using the minimum amount of information from <*K spec*> required to distinguish x from the other entities in the context, using the entity's substance as the value for <*R sem spec type category*>; the attribute <*R sem spec agr countable*> is set to '−', and <*R sem spec agr number*> is set to *sg*. Thus, subsequent references to entities like

(5-178) a ring of onion

are of the form

(5-179) the onion

where this is a mass, rather than a count, noun phrase.

If the <*K structure*> is *mass*, we build the substructure addressed as <*R sem spec type*> using the minimum amount of information from <*K spec*> required to distinguish x from the other entities in the context, using the entity's substance as the value for <*R sem spec type category*>; the attribute <*R sem spec agr countable*> is set to '−', and <*R sem spec agr number*> is set to *sg*. Thus, even if quantity information is contained within the knowledge base structure, this is ignored when generating subsequent references.

If the <*K structure*> is *set*, and the constituents of the set are not explicitly listed, we build the substructure addressed as <*R sem spec type*> using the minimum amount of information from <*K spec element*> required to distinguish x from the other entities in the context, using the substance of the entity's elements as the value for <*R sem spec*

type category>; the attribute <*R sem spec agr countable*> is set to '+', and <*R sem spec agr number*> is set to *pl*. Thus, even if quantity or cardinality information is contained within the knowledge base structure, this is ignored when generating subsequent references.

If the <*K structure*> is *set*, and the constituents of the set are explicitly listed, we first try build the substructure addressed as <*R sem spec type*> using a superordinate term which describes all the elements of the set. If this is not possible, then the current algorithm is applied to each the knowledge base structure corresponding to each element of <*K spec constituents*>, with the results being accumulated in as a list in <*R sem spec type*>.

In each case above, if a unique description can be constructed, then <*R sem status unique*> is set to '+'; otherwise it is set to '−'.

5.6 One-Anaphora

In this section, we turn to the generation of *one*-anaphoric expressions. First, we consider the scope for viewing *one*-anaphora as being purely a matter of syntactic substitution, before going on to present the approach taken in EPICURE, where *one*-anaphora takes place in the mapping from recoverable semantic structure to abstract syntactic structure.

5.6.1 The Syntactic Analysis of One-Anaphora

Recall, from chapter 2, that there are two kinds of what we might call *one*-anaphora. First, we have the words *one* and *some*, which seem to function as pro-forms for complete noun phrases, as in the following examples:

(5-180) a Do you have any green peppers?
b Yes, I have <u>one</u>.

(5-181) a Do you have any green peppers?
b Yes, I have <u>some</u>.

However, *one* can also, along with *ones*, function as a pro-form for a nominal expression

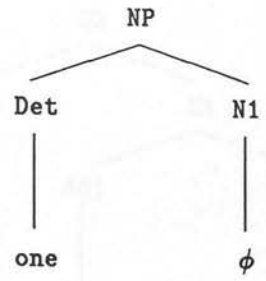


Figure 5.20: *One* as a determiner

or N1 category. Thus, we have

(5-182) a Do you have any green peppers?
b No, but I have a red one.

(5-183) a Do you have any green peppers?
b No, but I have some red ones.

In the case of the *one/some* pair, it seems most plausible to suggest that the forms are not actually substituting for complete NPs, but rather are functioning as determiners in situations where the remainder of the containing noun phrase has been elided in each case. Such an analysis would then also explain instances like (5-184b) without the need to postulate that all the cardinal numbers are pro-NPs:

(5-184) a Do you have any green peppers?
b Yes, I have sixteen ϕ.

Accordingly, then, we suppose the surface syntactic structure of the *one* (5-180b) to be that shown in figure 5.20,[23] and the surface syntactic structure of the noun phrase *a red one* in (5-182b) to be that shown in figure 5.21.

[23]Strictly speaking, to be consistent with the analyses presented earlier in this chapter, *one* here would be of category Q rather than Det, but we ignore this for the purposes of the present exposition.

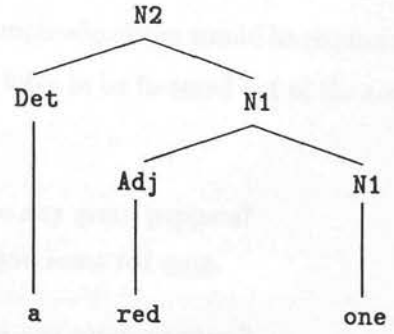


Figure 5.21: *One* as a pro-N1 form

5.6.2 One-Anaphora as Syntactic Substitution

Focusing first on the *one/ones* pair, we might take the view that the word *one* literally substitutes for N1 syntactic constituents. Suppose the preceding context contains the noun phrase *a green pepper* and we are about to generate a reference to an entity where the semantic content of that reference is to be the same as that in the noun phrase *a red pepper*. In order to generate an instance of *one*-anaphora here, we could use the following algorithm, where NP_1 is the noun phrase in the preceding context:

- construct the complete syntactic structure for the required semantic content;
- compare this structure with that of NP_1: if they have any N1 constituents in common, find the largest and replace this by the pro-N1 form *one*.

Thus, in the present example, we would first construct the surface syntactic structure corresponding to *the red pepper*, then recognize that the N1 constituent *pepper* occurs in both, and replace it with *one* in the noun phrase being generated, resulting in the noun phrase *a red one*.

Because of the recursive structure of N1 constituents, this simple algorithm deals straightforwardly with examples like the following:

(5-185) a Helen bought a large old Germanic manuscript.
b Marc could only afford a small one.

Some augmentation of the simple algorithm would be required to make it work properly. For example, number would have to be factored out of the comparison of N1 structures, to cover the following data:

(5-186) a Do you have any green peppers?
b No, but I have some red ones.

(5-187) a Do you have any green peppers?
b No, but I have a red one.

(5-188) a Do you have a green pepper?
b No, but I have some red ones.

Similarly, the algorithm would have to be restricted to count noun phrases, to rule out the following:

(5-189) a Do you have some white rice?
b *No, but I have a brown one.

However, there are other problems. First, it is not clear how the rule can be elegantly integrated with a rule for the use of *one/some*: given the syntactic analyses used here, the resulting rule would be something like the following (again, modulo appropriate augmentations to deal with the issues of number and countability just mentioned):

- construct the complete syntactic structure for the required semantic content;
- compare this structure with that of NP_1: if they have any N1 constituents in common, find the largest and replace this by the pro-N1 form *one*, unless the largest common N1 is immediately dominated by the NP node in the new noun phrase, in which case elide the N1 altogether.

Even this is not enough, since we then require some mechanism to ensure that the determiner *a* is replaced by *one*.

Although a little inelegant, a rule with the above behaviour could be specified. Other problems, however, make this approach increasingly less appealing. First, it does not rule out the following:

(5-190) a Do you have any wine bottles?
b No, but I have a red one.

since, *syntactically*, *one* can stand in for *wine bottle*. This suggests that the substitution should be semantically constrained. In fact, this is essentially the approach taken in the most sophisticated approach to the generation of *one*-anaphora to be found in the literature: in the HAM-ANS system [Jameson and Wahlster 1982], the possibilities for *one*-anaphora are determined by considering a structure which corresponds to our abstract syntactic structure; successively smaller subtrees in this structure are considered for ellipsis until one is found that can be elided without information loss. Since the structures being compared are essentially semantic in nature, [24] it is reasonable to suppose that examples like (5-190b) could be ruled out relatively straightforwardly.

Apart from this benefit, carrying out the substitution at the level of semantics has other advantages: the idea of generating an entire parse tree and then replacing parts of it seems both computationally inefficient and psychologically unreal. However, the approach just described suffers from another problem, precisely because the semantic representation in question is so close to the surface syntactic representation. This only becomes apparent when we consider noun phrases which have postmodifiers as well as premodifiers. Consider the following example:

(5-191) a Do you have a large yellow brick made of gold?
b No, but I have a small *one*.

Here, *one* appears to be standing in for *yellow brick made of gold*; and so, we might assume the surface syntactic structure of *a large yellow brick made of gold* to be as in figure 5.22 (this has been simplified a little to omit detail irrelevant for the current purposes).

However, note that we can also have the following:

(5-192) a Do you have a large yellow brick made of gold?
b No, but I have a small *one* made of silver.

Now, in order to be able to generate this instance of *one*-anaphora by means of the approaches described above, we would need to attribute a *different* surface syntactic

[24] Our abstract syntactic structure also encodes information of a semantic nature, as we noted in section 4.4; in this respect, it is very similar to the structure used by Jameson and Wahlster.

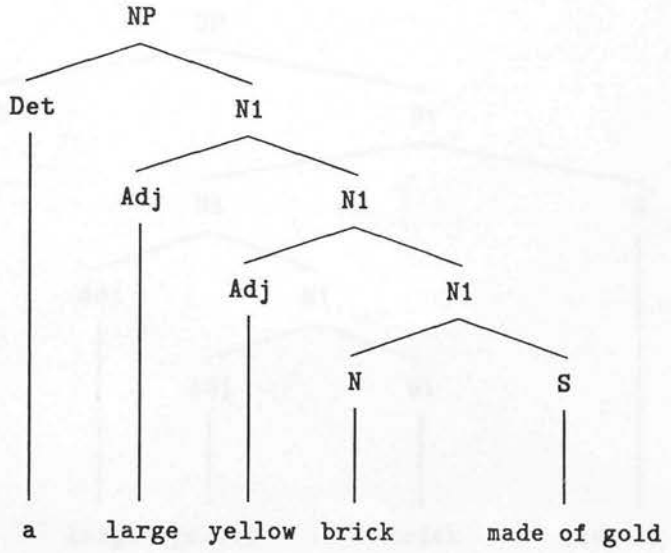


Figure 5.22: One possible analysis of *a large yellow brick* ...

structure to the noun phrase *a large yellow brick made of gold*, as shown in figure 5.23.

There is no single structural analysis of the original noun phrase makes available all the possible antecedents for *one*-anaphors as syntactic constituents; and this is true no matter what syntactic analysis we use. In fact, the only way to make the possible antecedents available structurally would be to assign the noun phrase some kind of lattice-like structure.

This problem arises because the representations suggested match the surface syntax too closely. In the next section, we consider an alternative approach to *one*-anaphora which does not suffer from this restriction.

5.6.3 One-Anaphora and Recoverable Semantic Structure

In this thesis, we view *one*-anaphora as being an operation applied at the level of recoverable semantics. Recall that, for a noun phrase like

(5-193) a large ripe banana

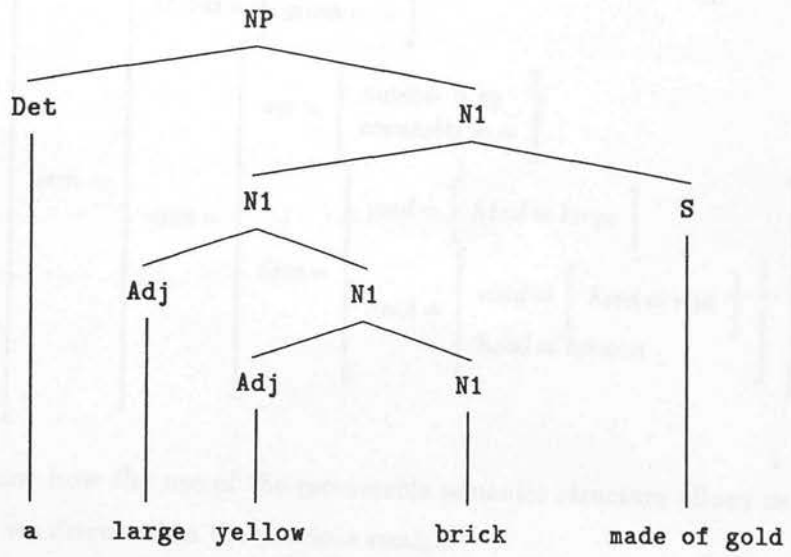


Figure 5.23: An alternative analysis of *a large yellow brick ...*

the corresponding recoverable semantic structure would be the following:

(5-194)

$$\left[\begin{array}{l} index = x_1 \\ sem = \left[\begin{array}{l} status = \left[\begin{array}{l} given = - \end{array}\right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = sg \\ countable = + \end{array}\right] \\ type = \left[\begin{array}{l} category = banana \\ properties = \left[\begin{array}{l} size = large \\ ripe = + \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

Thus, all the properties of the entity other than that which is to be realized by the head noun are maintained by means of what is essentially a flat list. In this respect, our approach is very close to that of Webber [1979].[25]. This contrasts with the abstract syntactic structure, which has the following form:

[25] Webber uses restricted quantification as a means of isolating the semantic content of the head noun of a noun phrase.

(5-195)
$$\left[\begin{array}{l} index = x_1 \\ sem = \left[\begin{array}{l} status = \left[\ given = - \ \right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = sg \\ countable = + \end{array}\right] \\ desc = \left[\begin{array}{l} mod = \left[\ head = large \ \right] \\ head = \left[\begin{array}{l} mod = \left[\ head = ripe \ \right] \\ head = banana \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

Below, we show how the use of the recoverable semantic structure allows us to avoid the problems we discussed in the previous section.

One Anaphora and the Discourse Model

First, it is important to note how the use of *one*-anaphora integrates with the use of the discourse model in EPICURE. The important point here is that it is the recoverable semantic structure of an utterance which is put in cache memory, and not the abstract syntactic structure. To see why this is so, consider the following sequence of utterances.

(5-196) a Irene bought a red t-shirt.
b Joe bought a blue <u>one</u>.
c Richard bought a red <u>one</u>.

One-anaphora appears to obey the same kind of locality constraint as does verb phrase ellipsis and most pronominalization; thus, we assume that the antecedent for a *one*-anaphor will always be found in cache memory. However, this means that, for the (5-196c) to be possible, the cache memory contents resulting from (5-196b) must include the semantic element corresponding to the noun *t-shirt*. This is not part of the abstract syntactic structure of the noun phrase *a blue one*, but it is part of the recoverable semantic structure corresponding to this noun phrase. For this reason, it is the recoverable semantic structure of an utterance that is saved in cache memory.

To see how the generation of *one*-anaphora works in EPICURE, consider the following short discourse:

(5-197) a Slice the green pepper.

b Now remove the top of the red one.

Just before generation of (5-197b) commences, that part of the cache memory structure corresponding to the noun phrase in (5-197a) will be as follows:

(5-198)
$$\left[\begin{array}{l} index = x_1 \\ sem = \left[\begin{array}{l} status = \left[given = - \right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = sg \\ countable = + \end{array}\right] \\ type = \left[\begin{array}{l} category = pepper \\ properties = \left[colour = green \right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

As the semantic content of (5-198b) is determined, the current clause work space will contain the following RS structure corresponding to the noun phrase:

(5-199)
$$\left[\begin{array}{l} index = x_2 \\ sem = \left[\begin{array}{l} status = \left[given = - \right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = sg \\ countable = + \end{array}\right] \\ type = \left[\begin{array}{l} category = pepper \\ properties = \left[colour = red \right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

For simplicity, we rule out *one*-anaphora if the clause in cache memory contains more than one participant (other than the agent). This removes the need to provide a more sophisticated mechanism for dealing with examples like the following:

(5-200) a Do you have a green pepper or a carrot?

b *Yes, I have one.

The possibilities for the use of *one*-anaphora are then determined by RS→AS mapping rules. In the following, RS_1 is the recoverable semantic structure corresponding to the noun phrase in the previous utterance, and RS_2 is the recoverable semantic structure corresponding to the current noun phrase semantics being constructed.

First, we have a rule which checks for the case where an entity has exactly the same type as an entity mentioned in the previous sentence (i.e, that situation where the *one/some* pair can be used).

(5-201) if $<RS_1\ sem\ spec\ type> = <RS_2\ sem\ spec\ type>$
then $<AS_2\ sem\ spec\ desc\ head> = \phi$

The resulting abstract syntactic structure would then be as follows:

(5-202)
$$\left[\begin{array}{l} index = x_1 \\ sem = \left[\begin{array}{l} status = \left[\ given = -\ \right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = sg \\ countable = + \end{array}\right] \\ desc = head = \phi \end{array}\right] \end{array}\right] \end{array}\right]$$

The grammar realizes this structure as the word *one*.

However, if the two TYPE structures are not the same, then the second *one*-anaphora mapping rule may apply:

(5-203) if $<RS_1\ sem\ spec\ type\ category> = <RS_2\ sem\ spec\ type\ category>$
then build the appropriate N1 substitution

If this second rule triggers, then a little more work needs to be done to construct the appropriate abstract syntactic structure. The procedure here is essentially similar to that used in the construction of the abstract syntactic structure for a noun phrase which does not involve *one*-anaphora (as discussed in section 5.2): a list L of the properties to be realized has to be constructed, mirroring the embedding of the adjectives in the resulting noun phrase. In addition, however, those attribute-value pairs which appear in $<RS_1\ spec\ type\ properties>$ are omitted from L, and the element corresponding to the head noun is set to ϕ. The abstract syntactic structure in the case of our present example is then as follows:

(5-204)
$$\begin{bmatrix} index = x_1 \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = + \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} number = sg \\ countable = + \end{bmatrix} \\ desc = \begin{bmatrix} mod = \begin{bmatrix} head = red \end{bmatrix} \\ head = \phi \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

This results in the generation of the noun phrase *the red one*.

Although the appropriate grammar rules are not implemented in the current version of EPICURE, the above mechanisms will also work for cases where the *one*-anaphor substitutes for both premodifiers and postmodifiers, as discussed in the previous section.

More Complex One-Anaphora

Although the above rules work satisfactorily where references to straightforward individuals are concerned, there are some other cases where an additional mechanism is required. We note below some of the relevant details for incorporating more complex instances of *one*-anaphora into the present framework.

Suppose, for example, we have an ingredient which is packaged in a non-default way, as might be described by the noun phrase *a ring of onion*. The recoverable semantic structure corresponding to this will be as follows:

(5-205)
$$\begin{bmatrix} index = x \\ sem = \begin{bmatrix} status = \begin{bmatrix} given = - \end{bmatrix} \\ spec = \begin{bmatrix} agr = \begin{bmatrix} countable = + \\ number = sg \end{bmatrix} \\ type = \begin{bmatrix} category = ring \\ props = \begin{bmatrix} size = regular \\ substance = onion \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Now, consider the following data:

(5-206) a a ring of onion
b one of potato
c two of potato

(5-207) a a ring of onion
b ?a cube of it

(5-208) a an onion ring
b a potato one

Thus, the PACKAGING of the ingredient is available as an antecedent for *one*-anaphora, but the SUBSTANCE is not.

Similarly, where entities are described as quantities of individuals, note that we have the following *one*-anaphoric possibilities:

(5-209) a 4lbs of green peppers
b 2lbs of red ones
c 4lbs of green peppers and 2lbs of red [ones]

(5-210) a two ounces of salt
b one of pepper

Thus, in the following structure,

(5-211) RS =
$$\left[\begin{array}{l} index = x \\ sem = \left[\begin{array}{l} status = \left[given = - \right] \\ spec = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = pl \\ countable = + \end{array}\right] \\ quant = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = 4 \\ countable = + \end{array}\right] \\ type = \left[category = pound \right] \end{array}\right] \\ element = \left[\begin{array}{l} agr = \left[\begin{array}{l} number = pl \\ countable = + \end{array}\right] \\ type = \left[category = pepper \right] \end{array}\right] \end{array}\right] \end{array}\right] \end{array}\right]$$

both TYPEs are *one*-anaphorically accessible.

The full details of the mechanisms required here, and their interactions with the other aspects of *one*-anaphora, remain to be worked out.

5.6.4 Summary

Ultimately, *one*-anaphora is possible whenever an entity is of the same type as an entity mentioned in the local context, although the notion of type required here is very broad: basically, any collection of attribute-value pairs which includes an attribute-value pair that can be realized by means of a head noun constitutes a type. This also means that there is an ordering on types: some are underspecified with respect to others, and less specified types subsume those which are more fully specified.

In the mechanism described above, these types are implicit: that is, whether or not two entities share the same type is calculated from the information known about those entities. Another approach, suggested by McDonald [1980], would be to actually maintain a lattice-like structure of types. This approach seems intuitively more appealing than the approach described above: rather than building a recoverable semantic structure which is subsequently pared down to produce the required abstract syntactic structure, we could then work outwards from the shared type of the two entities, adding the distinguishing properties. The problem is, of course, the computational expense of maintaining such an elaborate structure of types: given n attributes, there could be as many as 2^{n-1} types. The issues involved in the implementation of such a scheme are beyond the scope of the present work.

In summary, the view taken in this section has been to suggest that *one*-anaphora is best dealt with at some level of semantic representation. However, evidence from other languages may suggest that this is not correct, and that a more 'surfacey' approach may be necessary.[26] To see this, consider the Dutch words ***huis*** and ***woning***: both words correspond to the English word ***house***, but ***huis*** is neuter whereas ***woning*** is gendered. When these words serve as the linguistic antecedents of *one*-anaphoric forms, any adjectives that modify the *one*-anaphors adopt the gender information of the antecedent: thus, in (5-212b), the adjective ***kleine*** is gendered because ***woning*** is gendered, and in (5-212c), the adjective ***klein*** is neuter because ***huis*** is neuter.

[26] This was pointed out to me by Gerard Kempen.

(5-212) a Jan has a big house and Mary a small one.

b Jan heeft een grote woning en Marie een kleine.

c Jan heeft een groot huis en Marie een klein.

Thus, the process of generating *one*-anaphoric expressions requires access to syntactic information about gender which we would not expect to be present at the level of semantics: this suggests that an account of *one*-anaphora which operates purely at the level of semantics will ultimately fail.

5.7 The Referring Expression Algorithm Summarised

In the preceding sections of this chapter, we have presented in detail the mechanisms used to generate both initial and subsequent references to various kinds of individuals, masses and sets, including the use of pronominal anaphora and one-anaphora.

5.7.1 The Basic Algorithm

For each intended referent x, the clause generation process will have constructed a semantic structure R which specifies the index of the entity, and optionally some information regarding its status (i.e., whether or not it is specified as obligatory by the case frame used in the current clause being constructed).

Given this structure R, the the basic referring expression algorithm is as follows. First, we build the recoverable semantic structure.

- If x is the current centre, then add this information to $<R\ sem\ status>$ and set $<R\ sem\ spec\ type>$ to be ϕ.
- Otherwise, if x is present in the discourse model, build the recoverable semantics of a subsequent reference as described in section 5.5.
- Otherwise, construct the recoverable semantics for an initial reference, as described in chapter 5.3.

The abstract syntactic structure A can then be constructed as follows.

- If x is both the centre and non-obligatory, then omit it from the abstract syntactic structure corresponding to the clause.
- If x is the centre but is obligatory then do nothing: this will be realised as a pronoun.
- Otherwise, see if *one*-anaphora is possible, as described in section 5.6, and construct the abstract syntactic structure appropriately.
- If *one*-anaphora is not possible, then construct the abstract syntactic structure as described in sections 5.2 and 5.3.

5.7.2 The Noun Phrase Grammar

The complete noun phrase grammar as presented in this chapter is as follows.

$$
\begin{array}{lll}
NP \rightarrow Det\ N1 & & \\
\quad <Det\ sem> & = & <NP\ sem\ status> \\
\quad <NP\ syn\ agr> & = & <NP\ sem\ spec\ agr> \\
\quad <N1\ syn\ agr> & = & <NP\ syn\ agr> \\
\quad <Det\ syn\ agr> & = & <N1\ syn\ agr> \\
\quad <N1\ sem> & = & <NP\ sem\ spec\ desc> \\
 & & \\
NP \rightarrow Pronoun & & \\
\quad <Pronoun\ sem\ status> & = & <NP\ sem\ status> \\
\quad <Pronoun\ sem\ spec> & = & <NP\ sem\ spec\ desc> \\
\quad <Pronoun\ syn\ agr> & = & <NP\ sem\ spec\ agr> \\
 & & \\
NP_1 \rightarrow NP_2\ PP[+of] & & \\
\quad <NP_2\ sem\ status> & = & <NP_1\ sem\ status> \\
\quad <NP_2\ sem> & = & <NP_1\ sem\ spec\ desc\ spec> \\
\quad <PP\ sem> & = & <NP_1\ sem\ spec\ desc\ set> \\
 & & \\
NP_1 \rightarrow NP_2\ N1[+of] & & \\
\quad <NP_2\ sem> & = & <NP_1\ sem\ spec\ desc\ spec_1> \\
\quad <N1\ sem> & = & <NP_1\ sem\ spec\ desc\ spec_2> \\
\quad <N1\ syn\ agr> & = & <NP_1\ sem\ spec\ agr> \\
\quad <NP_2\ sem\ status> & = & <NP_1\ sem\ status>
\end{array}
$$

$NP \rightarrow Q\ N1$

$\langle Q\ sem\ status \rangle$	=	$\langle NP\ sem\ status \rangle$
$\langle NP\ syn\ agr \rangle$	=	$\langle NP\ sem\ spec\ agr \rangle$
$\langle N1\ syn\ agr \rangle$	=	$\langle NP\ syn\ agr \rangle$
$\langle Q\ syn\ agr \rangle$	=	$\langle N1\ syn\ agr \rangle$
$\langle Q\ sem\ number \rangle$	=	$\langle NP\ sem\ spec\ agr\ number \rangle$
$\langle N1\ sem \rangle$	=	$\langle NP\ sem\ spec\ desc \rangle$

$N1 \rightarrow N$

$\langle N\ sem \rangle$	=	$\langle N1\ sem\ head \rangle$

$N1_1 \rightarrow AP\ N1_2$

$\langle AP\ sem \rangle$	=	$\langle N1_1\ sem\ mod \rangle$
$\langle N1_2\ sem\ head \rangle$	=	$\langle N1_1\ sem\ head \rangle$

$N1_1 \rightarrow N1_2\ N1_3$

$\langle N1_2\ sem \rangle$	=	$\langle N1_1\ sem\ spec\ desc\ mod \rangle$
$\langle N1_3\ sem \rangle$	=	$\langle N1_1\ sem\ spec\ desc\ head \rangle$

$N1 \rightarrow N1_1, N1_2, \ldots, Conj\ N1_n$

$\langle N1_1\ sem \rangle$	=	$\langle N1\ sem\ first \rangle$
$\langle N1_2\ sem \rangle$	=	$\langle N1\ sem\ second \rangle$
...		
$\langle N1_n\ sem \rangle$	=	$\langle N1\ sem\ nth \rangle$

$Q \rightarrow Num$

$\langle Num\ sem \rangle$	=	$\langle Q\ sem \rangle$

$AP_1 \rightarrow Adv\ AP_2$

$\langle Adv\ sem \rangle$	=	$\langle AP_1\ sem\ mod \rangle$
$\langle AP_2\ sem\ head \rangle$	=	$\langle AP_1\ sem\ head \rangle$

$AP \rightarrow Adj$

$\langle Adj\ sem \rangle$	=	$\langle AP\ sem\ head \rangle$

Chapter 6

A Worked Example

All the central elements of the material presented in chapters 3, 4 and 5 have been implemented in a computer program (written in C-PROLOG running under UNIX). In this chapter, we present a worked example which demonstrates many of the mechanisms described in the previous chapters, and discusses some of the implementational detail.

In section 6.1, we present a fairly typical recipe chosen from a cookery book. We make a number of simplifications to this recipe in order to avoid having to deal with phenomena that are not central to the concerns of this thesis. These simplifications are explicitly categorised and noted, and the resulting target recipe presented. We show, alongside this, the results of an attempt to generate this target recipe using EPICURE.

In the remainder of the chapter, we present a walk-through of the various stages of processing involved in generating the target recipe. In section 6.2, we describe the various inputs to the system in terms of their actual implementation in PROLOG. In section 6.3, we describe the various stages of processing, and note some of the limitations and inadequacies of the present implementation. Some of these are purely implementational issues; others, however, have ramifications for the theory presented in earlier chapters.

6.1 The Target Recipe

As a basis for the target recipe to be described in this chapter, we will use the recipe for butter bean soup shown in figure 6.1, from Rose Elliot's *The Bean Book*.

CREAM OF BUTTER BEAN SOUP

4 oz butter beans
1 large onion
1 medium-sized potato
2 carrots
2 sticks celery
1 oz butter
$1\frac{1}{2}$ pints water or unsalted stock
$\frac{1}{2}$ pint milk
a bouquet garni: a couple of sprigs of parsley, a sprig of thyme and a bayleaf, tied together
4–6 tablespoons of cream
sea salt
freshly ground black pepper
grated nutmeg

Soak the butter beans, then drain and rinse them. Peel and chop the onion and potato; scrape and chop the carrots; slice the celery. Melt the butter in a large saucepan and add the vegetables; saute them for 7–8 minutes, but don't let them brown, then add the butter beans, water or stock, the milk and the bouquet garni. Simmer gently, with a lid half on the saucepan, for about 1.25 hours, or until the butter beans are tender. Remove the herbs, then liquidise the soup, stir in the cream and add the sea salt, freshly ground black pepper and nutmeg to taste. Reheat the soup, but don't let it boil. Serve each bowl sprinkled with croutons.

Figure 6.1: Butter Bean Soup

There are various aspects of this recipe which are not directly relevant to the issues discussed in this thesis. Below, we make a number of simplifications to the recipe, thus permitting us to focus on the particular phenomena of interest in the context of the present work.

6.1.1 Simplifying the Recipe

We categorise here all the simplifications made to the target recipe. This categorisation provides us with a systematic way of re-introducing the complexity of the original recipe

if we should choose to do so at some later point. The major simplifications adopted are as follows:

1. reduction of sentences to single clauses;
2. removal of conditions on actions;
3. removal of temporal expressions;

In addition, we make a number of more minor changes. Each of these simplifications and changes is described below in more detail.

In general, the simplifications introduced are necessary because of inadequacies in the underlying representation of objects and events, rather than inherent limitations in the approach taken to syntax.

Reduction to Single Clauses

Since this thesis is primarily concerned with the generation of referring expressions, little emphasis has been placed on the construction of anything other than simple sentence structures. Thus, we expand each multi-clause sentence in the original recipe to a series of single clause sentences. This requires the removal of some conjunctions from the text, and results in slight lessening of discourse coherence: where the conjunction is *and*, such a transformation seems to be meaning preserving, but removal of *but*, for example, could be construed as resulting in some loss of meaning. For the present purposes, this is of no great significance.

Removal of Conditions on Actions

Some actions specified in the recipe have conditions of various sorts associated with them: for example, *reheat the soup, but don't let it boil.* Representation of the information in such clauses is a needless complication from our point of view, so we will remove these from the recipe.[1]

[1]Indeed, the incorporation of the planning constructs that would be necessary in order to model conditions of this kind is non-trivial, and an area of active research: see Tate [1985].

Removal of Temporal Expressions

We are not particularly interested in expressions which indicate the durations of actions (e.g., *for 7–8 minutes*), nor with the use of words which indicate temporal ordering (e.g., *then*). All such expressions are therefore removed.

Other Minor Changes

There are a number of other changes we make to simplify the generation process. Some of these are of little significance, while others are due to the limitations of the theory presented in this thesis.

1. Although there are numerous *ad hoc* possibilities, it is not immediately clear how we might represent the expression *to taste*, in *add the sea salt ... to taste* in an elegant manner: we therefore remove it from the recipe. This is not significant given the concerns of the present work.
2. We are asked to melt the butter *in a large saucepan*. Since we have been focusing on the generation of references to the ingredients in recipes, we ignore this referring expression. Similarly, we simplify the simmering instruction, *simmer gently, with a lid half on the saucepan* by removing the *with* prepositional phrase. Such entities are essentially functionally defined: as discussed in chapter 3, the representation of functional perspectives is beyond the scope of the present work.
3. The adverb of manner is also removed from the simmering instruction, since it cannot be accommodated in the representation currently used for events.
4. We remove the final command regarding the serving of the soup, since this would require the representation of another object which is not an ingredient (i.e., the referent of *each bowl*).
5. The recipe allows for either *stock* or *water* to be used. As discussed at the end of chapter 3, the representation of disjunction is a difficult problem, and one that has not been addressed in the present work; thus, we remove both the disjunction in the ingredients list, and the reference to it in the body of the instructions.
6. The currently-implemented domain modelling mechanism is not capable of handling the removal of parts from entities: thus, we remove the instruction to remove

the bouquet garni from the soup. We also remove all references to this ingredient. Again, this is not crucial to the issue of referring expression generation, but is rather a limitation inherent in the simple planning framework adopted in the present work.

7. The grammar used by EPICURE does not currently cater for the generation of expressions of range, and so we replace the ingredient specified as *4–6 tablespoons of cream* by *4 tablespoons of cream*. A relatively simple extension to the current noun phrase generation mechanism would permit generation of ranges like these.

6.1.2 Target Output and Actual Output

The result of the above simplifications is the target recipe shown in figure 6.2; figure 6.3 shows EPICURE's version of the target recipe. Line numbers have been added to both to aid comparison.

EPICURE's output differs from the target recipe in a number of ways. We will examine these in detail in the remainder of this chapter, where we walk through the generation process.

6.2 The Ingredients of Recipe Generation

In order to generate a given recipe, EPICURE requires three basic sets of data structures:

- a set of knowledge base entities corresponding to the ingredients in the recipe to be described;
- a set of planning operators that describe the effects of carrying out the required actions, and that (where appropriate) stipulate how actions may be decomposed; and
- some specification of the hearer's capabilities.

In this section, we describe the PROLOG representation of each of these. Some understanding of PROLOG will be necessary for much of what follows.

1 4 oz butter beans
2 1 large onion
3 1 medium-sized potato
4 2 carrots
5 2 sticks celery
6 1 oz butter
7 $1\frac{1}{2}$ pints unsalted stock
8 $\frac{1}{2}$ pint milk
9 4 tablespoons of cream
10 sea salt
11 freshly ground black pepper
12 grated nutmeg

13 Soak the butter beans.
14 Drain and rinse them.
15 Peel and chop the onion and potato.
16 Scrape and chop the carrots.
17 Slice the celery.
18 Melt the butter.
19 Add the vegetables.
20 Saute them.
21 Add the butter beans, stock and milk.
22 Simmer.
23 Liquidise the soup.
24 Stir in the cream.
25 Add the sea salt, freshly ground black pepper and nutmeg.
26 Reheat the soup.

Figure 6.2: A Simplified Target

6.2.1 The Representation of Feature Structures in Prolog

Various aspects of the generation process described in the preceding chapters make crucial use of unification. This makes PROLOG a natural choice for implementation; however, PROLOG provides primarily for TERM UNIFICATION, where the number of arguments of a given predicate is fixed, with each being identified by its position in the structure in question. For our purposes, GRAPH UNIFICATION would be more appropriate, since this permits values to be identified by their labels (i.e., by attribute), and, most importantly from our point of view, permits structures to be extended as desired. Thus, following Eisele and Dörre [1986:551], we represent feature structures by

four ounces of butter beans
a large onion
a medium potato
two carrots
two sticks of celery
one ounce of butter
1.5 pints of unsalted stock
0.5 pints of milk
four tablespoons of cream
some sea salt
some freshly ground black pepper
some grated nutmeg

Soak, drain and rinse the butter beans.
Peel and chop the onion.
Peel and chop the potato.
Scrape and chop the carrots.
Slice the celery.
Melt the butter.
Add the vegetables.
Saute them.
Add the butter beans, the stock and the milk.
Simmer.
Liquidise the soup.
Stir in the cream.
Add the seasonings.
Reheat.

Figure 6.3: The Output Produced by EPICURE

open-ended lists of pairs:

(6-1) `[A1 = V1, A2 = V2, ..., An = Vn|_]`

where each A_i is an atomic attribute, and each V_i is the value associated with that attribute. A value may either be an atomic symbol, a term denoting a feature structure, or a list of values (notated by means of a '+' preceding the list of values).

Unification of two feature structures is then achieved by the following PROLOG predicates:

(6-2)
```
merge(X, X):- !.
merge([A = V|Rest1], F2):-
      delete(A = V2, F2, Rest2),
      merge(V1, V2),
      merge(Rest1, Rest2).

delete(F, [F|X], X):- !.
delete(F, [A|X], [A|Y]):-
      delete(F, X, Y).
```

Thus, unification of two structures amounts to inserting into each of the two structures those that are present in the other; where both structures have a complex value for an attribute, these values must be recursively unified.

6.2.2 The Ingredients

The ingredients in a recipe are represented by a set of knowledge base structures. As discussed in chapter 3, although a state is essentially a set of propositions that describe what is true in that state, it is more convenient to model the behaviour of the ingredients in a recipe using a more object-centred approach: thus, each knowledge base structure describes a particular entity in a particular state. In the context of our target recipe, suppose our initial state is s_0: we then have the following knowledge base structure corresponding to the working set.

(6-3)
$$\left[\begin{array}{l} index = x \\ state = s_0 \\ spec = \left[\begin{array}{l} structure = set \\ constituents = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}] \end{array}\right] \end{array}\right]$$

Using the representation described in the previous section, the working set is represented in PROLOG by the following structure:[2]

```
(6-4)  kb([index = x1,
           state = s0,
           spec  = [structure    = set,
                    constituents = +[x1,x2,x3,x4,x5,x6,
                                     x7,x8,x9,x10,x11,x12]
                    |_]
           |_]).
```

Each constituent of the working set is then represented by a distinct knowledge base structure. So, for example, *a large onion* is represented thus:

$$\text{(6-5)}\quad \text{KB} = \left[\begin{array}{l} index = x_2 \\ state = s_0 \\ spec = \left[\begin{array}{l} structure = individual \\ substance = onion \\ packaging = \left[\begin{array}{l} shape = onion \\ size = large \end{array}\right] \end{array}\right] \end{array}\right]$$

Or, in PROLOG:

```
(6-6)  kb([index = x2,
           state = s0,
           spec  = [structure = individual,
                    substance = onion,
                    packaging = [shape = onion,
                                 size  = large
                                 |_]
                    |_]
           |_]).
```

The PROLOG representations of several different kinds of knowledge base entities are presented below.

A set such as that described by the noun phrase *two sticks of celery* is represented as follows:

[2]The + symbol at the head of the value of the constituents attribute is used to indicate to the system that this is a list-valued feature.

```
(6-7)  kb([index = x5,
           state = s0,
           spec  = [structure   = set,
                    cardinality = 2,
                    element     = [structure = individual,
                                   packaging = [shape = stick,
                                                size  = regular
                                                |_],
                                   substance = celery
                                   |_]
                    |_]
           |_]).
```

4 oz of butter beans is represented as follows:

```
(6-8)  kb([index = x1,
           state = s0,
           spec  = [structure = set,
                    quantity  = [number = 4,
                                 unit   = ounce
                                 |_],
                    element   = [structure = individual,
                                 packaging = [shape = butter_bean,
                                              size  = regular],
                                 substance = butter_bean
                                 |_]
                    |_]
           |_]).
```

The mass *1 oz of butter* is represented as

```
(6-9)  kb([index = x6,
           state = s0,
           spec  = [structure = mass,
                    substance = butter,
                    quantity  = [number = 1,
                                 unit   = ounce
                                 |_]
                    |_]
           |_]).
```

and the simple mass *grated nutmeg* as

```
(6-10)  kb([index = x12,
            state = s0,
            spec  = [structure = mass,
                     substance = nutmeg,
                     grated    = +
                     |_]
            |_]).
```

6.2.3 The Plan Operators

As discussed in chapters 3 and 4, a planning operator specifies the effects of an action, and the possible decomposition of an action. Within EPICURE, a planning operator is basically an underspecified event, such that when the participants in the event and the begin and end states of the event are instantiated, the result is a more fully specified event. In principle, the various kinds of information associated with a planning operator could be maintained within a single structure; however, for implementational reasons, the representation of operators is distributed, so that corresponding to each operator there are up to three distinct clauses:

- an `opschema` clause, which specifies the number and nature of the participants in the event;
- an `effects` clause, which specifies the effects of the operator; and
- an optional `constituents` clause, which specifies the decomposition of the action corresponding to the operator.

Thus, the ADDING operator has the following `effects` clause:

```
(6-11)  effects([index  = E,
                 occurs = [begin = S0,
                           end   = S1
                          |_],
                 spec = [substance    = adding,
                         participants = [in  = [base     = X1,
                                                addendum = X2
                                               |_],
                                         out = [result   = X3
                                               |_]
                         effects =
                          [add = +[:X3:spec:structure    = set,
                                   :X3:spec:constituents = +[X1,X2]],
                           delete = +[]
                          |_]
                        |_]
                |_]).
```

No possible decompositions are specified for the `adding` operator, since it is considered a primitive in the present system. The `bean_preparing` operator, however, has the decomposition as shown in figure 6.4.

```
constituents([index  = E,
  occurs = [begin = S0,
            end   = Sn|_],
  spec   = [substance    = bean_preparing,
            participants = [in  = [object = X|_],
                            out = [result = X|_]
                           |_],
            constituents = +[[index  = E1,
                              occurs = [begin = S0,
                                        end   = S1|_],
                              spec = [substance    = soaking,
                                      participants = [in  = [object = X
                                                             |_],
                                                      out = [result = X
                                                             |_]
                                                      |_]|_]|_],
                             [index  = E2,
                              occurs = [begin = S1,
                                        end   = S2|_],
                              spec = [substance    = draining,
                                      participants = [in =  [object = X
                                                             |_],
                                                      out = [result = X
                                                             |_]
                                                      |_]|_]|_],
                             [index  = E3,
                              occurs = [begin = S2,
                                        end   = Sn|_],
                              spec = [substance    = rinsing,
                                      participants = [in =  [object = X
                                                             |_],
                                                      out = [result = X
                                                             |_]
                                                      |_]|_]
                             |_]]
                    |_]
  |_]).
```

Figure 6.4: Bean Preparing

When an event is decomposed into a number of sub-events, the planning mechanism ensures that the appropriate ordering relations are maintained between the states. In the case of the bean_preparing operator, for example, when this structure is instantiated, the particular instantiations used for the begin and end states are asserted in the PROLOG database as being ordered appropriately: i.e.,

(6-12) S0 < S1 < S2 < Sn

6.2.4 The Hearer's Knowledge

The final kind of information EPICURE requires in order to be able to construct an appropriate discourse is some indication of the hearer's capabilities. In the context of the current recipe, we have the following:

(6-13)
```
physically_capable(user, adding).
physically_capable(user, chopping).
physically_capable(user, draining).
physically_capable(user, peeling).
physically_capable(user, liquidising).
physically_capable(user, melting).
physically_capable(user, reheating).
physically_capable(user, rinsing).
physically_capable(user, sauteing).
physically_capable(user, scraping).
physically_capable(user, simmering).
physically_capable(user, slicing).
physically_capable(user, soaking).
physically_capable(user, stirring_in).

physically_capable(epicure, requesting).
```

Note that changing each of these data structures will have different effects on the output generated by EPICURE. Of course, the particular ingredients and plan operators provided determine the particular recipe that EPICURE will generate. However, changing the system's knowledge what the hearer is 'physically capable' determines how that particular recipe will be described; also, as we will see, making different assumptions about the underlying structure of the recipe plan can have quite noticeable effects on the eventual output.

6.3 Generating the Recipe

In this section, we describe how EPICURE uses the various inputs described in the previous section to generate a recipe.

6.3.1 The Top Level Goal

In conjunction with the data structures described in the previous section, EPICURE is given a top level goal corresponding to the recipe to be generated. In a full-blown planning system, we would present the planner with an initial state and a goal state, and have the system work out a plan to perform the transition between the two. Currently, EPICURE does not incorporate a planner appropriate to this task, and so, instead, we present EPICURE with a top level goal for which it already knows an appropriate decomposition. The planning axioms are then used to determine to what extent the event corresponding to this goal should be decomposed to accommodate the hearer's particular capabilities.

In the case of the target recipe, the top level goal is a making butter bean soup event, whose decomposition is defined by the `constituents` clause shown in figure 6.5.

6.3.2 The Planning Axioms

The planning axioms are essentially identical to those specified in chapter 4, except that they also return a suitable plan. The first clause determines whether an agent is capable of an action, and if he or she is, instantiates the agent slot of that event appropriately:

(6-14)
```
can_guarantee(Agent, Eventuality):-
        path_value(spec:substance, Eventuality, Action),
        physically_capable(Agent, Action),
        effects(Eventuality),
        makedag([:spec:participants:agent = Agent],
                Eventuality, Eventuality).
```

The `path_value` predicate used here determines the value of the specified path in the specified feature structure, and the `makedag` predicate instantiates a feature structure in accordance with a list of one or more constraints.

```
constituents([index  = Index,
  occurs = [begin = S0,
            end   = Sn|_],
  spec   = [substance    = butter_bean_soup_making,
            participants = [in  = [object = +[A,B,C,D,E,F,G,H,I,J,K,L]
                                   |_],
                            out = [result = X
                                   |_]
                           |_],
            constituents =
                +[[index  = I1,
                   occurs = [begin = S0,
                             end   = S1
                            |_],
                   spec   = [substance    = preparing_bb_ingredients,
                             participants = [in  = [object = +[A,B,C,D,E]
                                                    |_],
                                             out = [result = +[A,B,C,D,E]
                                                    |_]
                                            |_]
                            |_],
                  [index  = I2,
                   occurs = [begin = S1,
                             end   = Sn
                            |_],
                   spec   = [substance    = cooking_bb_soup,
                             participants =
                                 [in  = [object = +[A,B,C,D,E,F,G,H,I,J,K,L]
                                         |_],
                                  out = [result = X
                                         |_]
                                 |_]
                            |_]]
        |_]
  |_]).
```

Figure 6.5: The Top Level Decomposition

The delegation strategy is implemented very straightforwardly:

```
(6-15)  can_guarantee(epicure, Eventuality):-
                physically_capable(epicure, requesting),
                can_guarantee(user, Eventuality).
```

Finally, we have the action decomposition strategy:

```
(6-16)  can_guarantee(Agent, Eventuality):-
                constituents(Eventuality),
                path_value(spec:constituents, Eventuality,
        +SubEvents),
                can_guarantee1(Agent, SubEvents).
```

The `can_guarantee1` predicate checks that each element of the list of `SubEvents` can be guaranteed. The recursion here terminates when a level of decomposition is reached where the hearer is believed to be capable of carrying out the actions required. The resulting plan structure is then returned as the value of `Eventuality`.

Given the hearer's capabilities specified in the previous section, applying this mechanism to the top level goal results in the plan structure shown in figure 6.6; the key provides an English gloss of each node in the plan, with those nodes which are not explicitly described in the output surrounded by square brackets.

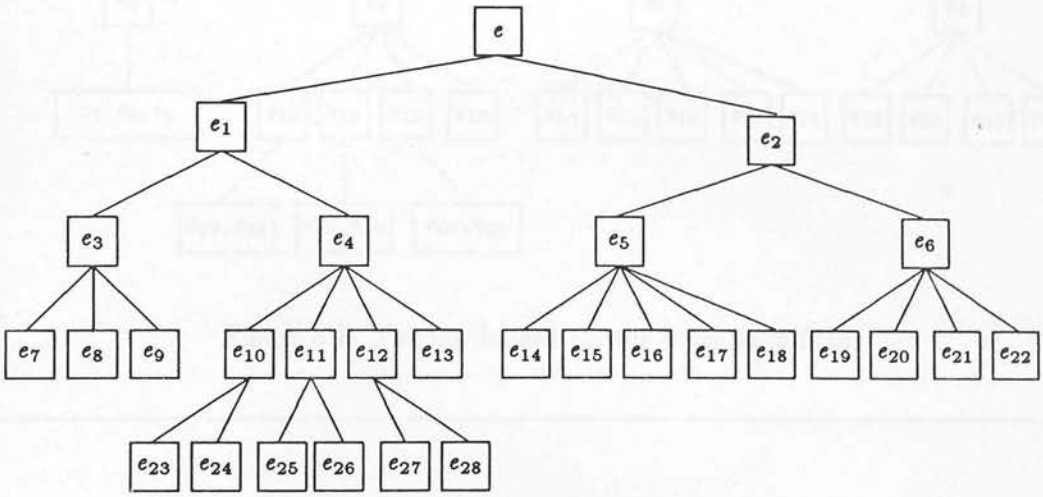
6.3.3 Optimisation

The optimisation mechanisms are then applied to this plan, resulting in some events being folded together. As discussed in chapter 4, the only optimisations implemented in the current version of the system are the collapsing together of those events which have the same arguments or the same operator.

The resulting structure is that shown in figure 6.7.

6.3.4 Discourse Generation

The plan is then passed to the discourse generator, which walks over it as specified in chapter 4, using a depth-first recursive descent strategy. The discourse model is maintained appropriately during this process, with new focus spaces being opened as



e	[Make butter bean soup]		
e_1	[Prepare the ingredients]	e_{15}	Add the vegetables
e_2	[Cook the ingredients]	e_{16}	Saute the vegetables
e_3	[Prepare the beans]	e_{17}	Add the beans, stock and milk
e_4	[Prepare the vegetables]	e_{18}	Simmer
e_5	[First stage of cooking]	e_{19}	Liquidise the soup
e_6	[Second stage of cooking]	e_{20}	Stir in the cream
e_7	Soak the beans	e_{21}	Add the seasonings
e_8	Drain the beans	e_{22}	Reheat
e_9	Rinse the beans	e_{23}	Peel the onion
e_{10}	[Prepare the onion]	e_{24}	Chop the onion
e_{11}	[Prepare the potato]	e_{25}	Peel the potato
e_{12}	[Prepare the carrots]	e_{26}	Chop the potato
e_{13}	Slice the celery	e_{27}	Scrape the carrots
e_{14}	Melt the butter	e_{28}	Chop the carrots

Figure 6.6: The Butter Bean Soup Plan

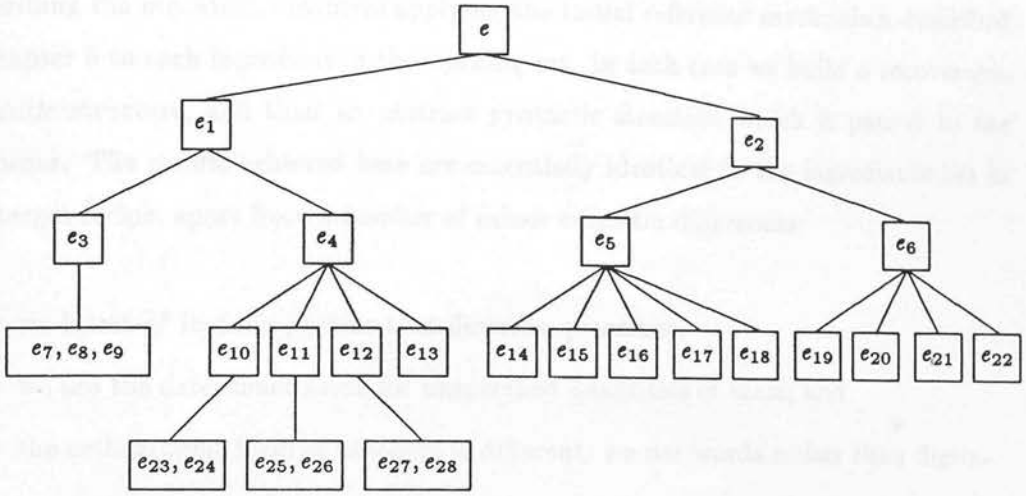


Figure 6.7: The Optimised Butter Bean Soup Plan

each subsequent level in the plan is reached, and being closed as the corresponding discourse segment ends.

The event specifications at the leaves of the structure are first passed to the domain modeller, and then to the clause generator, again as described in chapter 4. In the next section, we compare the target and the actual output of EPICURE, and describe some of the more interesting aspects of the generation of text that makes up this recipe.

6.4 Generating the Text

EPICURE's output differs from the target recipe in a number of ways. Below, we highlight various steps in the process of generating the target, commenting upon some points of interest.

6.4.1 Describing the Ingredients

Describing the ingredients involves applying the initial reference mechanism described in chapter 5 to each ingredient in the working set. In each case we build a recoverable semantic structure, and then an abstract syntactic structure which is passed to the grammar. The results achieved here are essentially identical to the ingredients list in the target recipe, apart from a number of minor cosmetic differences:

- we insert *of* in noun phrases that describe quantities;
- we use the determiner *some* for unspecified quantities of mass; and
- the orthographic form of numbers is different: we use words rather than digits.

The first two differences could be removed by adding to the system a parameter that specifies whether or not a 'telegraphic form' is used when building a noun phrase: note that, in general, we *do* want to include words like *of* and *some* when ingredients are referred to in the body of the recipe.

6.4.2 Describing the Recipe

More noteworthy differences occur in the body of the recipe itself.

Describing More Than One Operation in A Sentence

First, consider line 13 in the actual output, which corresponds to lines 13 and 14 in the target output. In the actual output, the the operations of *soaking*, *draining* and *rinsing* are collected together by the optimisation phase:

(6-17) Soak, drain and rinse the butter beans.

This is possible because all three actions are subactions in the decomposition of the *bean-preparing* operator: recall from chapter 4 that the optimisation routines operate across daughters that have the same mother node in the plan (where those daughters are atomic events). In the target recipe, however, only the second and third actions are collected together:

(6-18) Soak the butter beans.
Drain and rinse them.

Note that, by modifying the plan decomposition, and therefore the discourse structure, we can produce the same configuration as that found in the target. We can add a further intermediate level of decomposition, such that bean-preparation has two stages, where the first stage consists only of a *soaking* action, and the second stage consists of the two subactions of *draining* and *rinsing*. Since the optimisation routines *only* operate across daughters that have the same mother node, we would then generate exactly the description found in the target.

The operation of the optimisation routines gives rise to a related difference found in lines 13 and 14 of the actual output, which corresponds to line 14 of the target: whereas we generate

(6-19) Peel and chop the onion.
Peel and chop the potato.

in the target these operations are collected together to produce

(6-20) Peel and chop the onion and potato.

One way we could produce this output would be to modify the optimisation routines so that they operate on elements which are not atomic: however, notice that this then produce the wrong results in the case of the *soaking, draining and rinsing* discussed above. An alternative solution would be to modify the plan structure so that, instead of there being a level of structure intermediate between the plan node corresponding to *preparing the vegetables* and the more specific actions described, these specific actions would be daughters of the *preparing the vegetables* node.

What both these differences emphasise, however, is the arbitrariness involved in proposing that a particular structure underlies a piece of text. This is related, as we saw in chapter 3, to the question of the individuation of events. Finding satisfactory answers to these questions is, of course, beyond the remit of the present work.

Note that, in many cases, a participant in an event is omitted from the description of that event. This occurs in lines 19, 21, 22, 24, and 26 in both the target and the actual output: these are repeated below, with the omitted participant added in brackets.

(6-21) a Add the vegetables [to the butter].
b Add the butter beans, the stock and the milk [to the sautéed vegetables].
c Simmer [the vegetables, butter beans, stock and milk].
d Stir in the cream [to the soup].
e Add the seasonings [to the soup].

In addition, note that in the actual output, line 26 is generated as

(6-22) Reheat.

whereas in the target recipe, line 26 is

(6-23) Reheat the soup.

Recall from chapters 4 and 5 that, within EPICURE, a participant can be omitted from the description of an event if that participant is marked as being both non-obligatory and also as being the current centre. Whether or not the description of a participant is obligatory is determined by the case frame of the associated verb; and the centre of an utterance is, in the current domain, the result of the preceding operation.

This mechanism generates (with the exception of line 26) the desired output, but at best it is a gross simplification of what is really going on. In particular, consider lines 19 and 20:

(6-24) Add the vegetables.
Sauté them.

Just as in the target recipe, we generate the appropriate pronoun. Note that the result of the adding operation here is, strictly speaking, butter and vegetables combined; thus, we generate a pronoun because we apply the sauteing operation not to the result of the previous operation, but to one of its inputs.[3] Although we get the correct results, the

[3]Recall that our pronominalization algorithm permits a pronominal reference to be used to any entity mentioned in the previous sentence.

method is a little dubious: there ought to be a more principled way to make the butter 'disappear'. One solution would be to distinguish 'central' ingredients from this which play an 'in-service' role: the details of how this might work are left for future work.

Returning to the case of *reheat the soup*, here EPICURE elides the reference to the soup although the target recipe does not. In fact, a simple modification to the plan structure will produce the desired result. First, note why EPICURE elides the reference to the soup. The *reheating* event is a sister event to the preceding *add* event: since the *reheating* event is applied to the result of the *adding* event, the soup is the centre, and since the case frame for the verb *reheat* marks its object as non-obligatory, the mechanisms that build the abstract syntactic structure can omit the reference to the soup. However, recall that centres are retained across discourse segment closures: thus, if we restructure the plan so that the *reheating* action is not a sister of the preceding *liquidising*, *stirring* and *adding* actions, but is in a separate discourse segment, then the object of the reheating action must be described by a definite noun phrase. This is exactly what happens in line 23:

(6-25) Liquidise the soup.

where the soup is explicitly mentioned even though it is the result of the previous operation, because this action is in a distinct discourse segment to that which contains the previous operation.

Describing Mixtures

One problem for which EPICURE does not provide an elegant solution is that of describing the results of collecting different ingredients together (as distinct from describing a collection of similar ingredients: see below). In the context of the present recipe, the question is this: at which point in the recipe do we have *soup* rather than just a collection of ingredients?

Notice that the first mention of the soup is in line 23:

(6-26) Liquidise the soup.

In fact, however, the entity referred to as the soup is actually introduced previous to

this, in line 21:

(6-27) Add the butter beans, the stock and the milk.

Although not mentioned in this or in the following sentence (because it is then elided as discussed above), a distinct entity corresponding to the soup, with the butter beans, stock, milk and vegetables as constituent objects, is introduced as the result of this *adding* operation. In order to enable this to be described as *the soup*, however, we make use of a special form of the *adding* operator, which has the effect that the object which is the result of the *adding* operation has as its substance *soup*. This is far from elegant, but it is difficult to see how this could be done in a more principled way without a sizeable increase in the complexity of modelling the execution of the recipe.[4]

Using Superordinate Terms

Finally, we should note the use of superordinate terms in both lines 19 and 25:

(6-28) a Add the vegetables.
b Add the seasonings.

The first of these is as in the target recipe; however, the target does not use a superordinate in line 25. EPICURE uses a superordinate here by virtue of the following information in the knowledge base

(6-29)
```
ako(pepper, seasoning).
ako(salt, seasoning).
ako(nutmeg, seasoning).
```

and by virtue of the assumption that the hearer knows these facts. Thus, we could generate the target text here by abandoning this assumption; or, perhaps more plausibly, by not looking upon nutmeg as being the same kind of thing as salt and pepper.

[4]Ideally, we would want to reason from what is known about the constituent ingredients at the time the *adding* event takes place: so, for example, the fact that the ingredients are hot and that they include an appropriate quantity of liquid warrants the use of the name *soup*.

Chapter 7

Conclusions and Future Directions

In this chapter, we consider the extent to which the aims of the thesis stated in chapter 1 have been met, before going on to discuss briefly a number of issues that could be pursued further on the basis of the present work.

7.1 The Aims Revisited

The particular technical aims of this thesis, as discussed in chapter 1, can be summarised as follows:

- to present a collection of algorithms and data structures for the generation of pronouns, anaphoric definite noun phrases, and *one*-anaphoric phrases, by consolidating and advancing upon what has been done in this area by earlier researchers.

After a survey of the issues that arise in reference (chapter 2), and a close analysis of the particular kinds of referring expressions that appear in a particular domain, we presented a knowledge representation language that embodied all the relevant distinctions required by the ontology we adopted (chapter 3). We incorporated this ontology into a wider framework for language generation as a whole (chapter 4), and showed how the knowledge representation could be successfully used to produce appropriate referring expressions for a range of complex object types. In so doing, we also elaborated algorithms for the generation of various kinds of subsequent reference which paid heed to

the complexity of the objects to be described (chapter 5).

Although there are obviously gaps in the coverage of the work presented, and limitations to the mechanisms proposed, the originally stated aim of the work has been met. In the course of doing so, we have touched on a number of different areas. Amongst the various ideas explored in the present work, the following are the most important.

- We introduced the the notion of a generalized physical object as a way of representing singular entities, mass entities, and entities which are sets. The representation also allows parallels to be drawn between objects and events, although these parallels are not explored in the present work.
- We adopted the view that planning operators are essentially underspecified events, and used this, in conjunction with a simple model of the hearer, to allow us to determine the appropriate level of detail at which a given plan should be described.
- We made use of discourse model that distinguished local and global focus; the model adopted was closely tied to a notion of discourse structure, which allowed us to make a specific claim about long-distance pronominalization. There is considerable scope for further work in testing some of the claims made for the effects of discourse structure on the form of referring expressions.
- We presented a model of the generation of referring expressions that made use of two intermediate levels of representation; apart from simplifying the processing conceptually, this allowed us to suggest an approach to the generation of *one*-anaphora which overcomes the deficiencies of previous attempts.
- We integrated this model with the use of a linguistically-founded grammar for noun phrases, and suggested a number of extensions that could be added to the present framework relatively straightforwardly.
- We introduced a notion of DISCRIMINATORY POWER as a means to choosing the content of a referring expression.

Of course, there is much more to be done in all of these areas: indeed, I hope to have shown that the framework is a powerful one.

In chapter 1, we also noted more general aims that might be set for a natural language generation system. The 1982 ACL panel on text generation [Mann *et al* 1982] attempted

to assess the state of the art at that time, and to identify the kinds of technical developments that would be required over the next decade. Four particular components were identified as being necessary for a competent text generation facility:

- a comprehensive, linguistically justified grammar;
- a knowledge representation formalism that can encode diverse kinds of representation;
- a model of the intended reader of the text; and
- a model of discourse structure and control.

The work described in the present thesis has addressed each of these issues to some extent.

Most central are the first two. The noun phrase grammar used, although limited in certain respects, provides a reasonably wide coverage of noun phrase structures, and adheres to the now commonplace unification grammar approach. Its linguistic well-foundedness is due primarily to the influence of GPSG [Gazdar, Klein, Pullam and Sag 1985]. The knowledge representation formalism used is very general, but is particularly oriented towards the needs that arise from a desire to represent a complex ontology. It can be viewed as one layer of a multi-level knowledge representation language that permits encoding of different kinds of information.

The other two issues have also been addressed, although to lesser extents. EPICURE uses a hearer model, albeit limited, primarily in order to determine what level of detail should be used in describing a plan of action. This approach obviously has application in a wide range of instructional domains. Finally, the system makes explicit use of discourse structure, both in planning a discourse and in realizing it.

Of course, there is still a great deal to be done in each of these areas. Some possible directions that could be taken on the basis of the present work are considered in the next section.

7.2 Future Directions

At the beginning of this thesis, we expressed the view that the way forward for work in language generation required research to be focused on particular issues. The present work has focused on the generation of referring expressions as a particular issue, but in so doing we have found it necessary to touch on many other issues that arise in the generation task. In this section, we mention some avenues of possible research for which the present work might serve as a basis.

7.2.1 Generating Referring Expressions

Although various aspects of the generation of referring expressions have been covered in detail in the present work, the problems are by no means solved. The mechanisms described in the preceding chapters possess a number of particular limitations which could fruitfully be addressed. Among the more interesting of these are the following.

- Although the ontology developed in the course of the present work is fairly sophisticated, it has its limitations. In particular, as discussed at the end of chapter 3, the ontology described here is perhaps best viewed as the bottom layer of a multi-layer knowledge representation formalism, where subsequent layers allow for the representation of different perspectives on objects. This would then permit the representation of the kinds of objects (such as blocks and boxes) found in systems which make use of simpler ontologies, but would allow us to do so with much stronger underpinnings. It remains to be seen what the ramifications of this would be for referring expression generation.
- A major limitation of the present work is its avoidance of any complexity of quantification. This limitation is acceptable in the domain of application used in the thesis, but should be removed to permit wider application of the knowledge representation formalism.
- Related to the above is the potential for further expansion of the grammatical coverage of the system, particularly with respect to determination.
- Another direction in which the mechanisms could be extended is with respect to other kinds of reference: in particular, the representation language could be

extended to deal with attributive, non-specific and generic reference. The work of Kronfeld [1988] would be of relevance in this respect.

- Finally, the mechanisms for anaphora generation could be extended to deal with other kinds of anaphora, particularly verb phrase ellipsis.

7.2.2 Generating Connected Discourse

Apart from these extensions, there are also a number of other areas which have been touched upon in the present work, and which could be pursued further. Two of particular interest are as follows:

- There is a wealth of issues to be explored in the relationship between discourse structure and discourse generation. To name but a few: what is the most suitable notion of discourse structure for use in discourse generation? How adequate is an account based purely on the *structural* aspects of a discourse? Do the *semantics* of the particular relations between the discourse segments have no effect? What size are discourse segments? Can all forms of reference be explained in structural terms, or do we still require a notion of linear distance? What is the relationship between speech act theory and discourse intention?
- The system described here involves generating natural language from plans. There are many other back-ends one might use to supply input to the generation process, but even within this single area there are many unresolved questions. In particular, the mechanisms could be extended to deal with non-linear planners, and to deal with more complex planning systems which permit iteration and the use of conditionals.

The present work has demonstrated at least a few of the ingredients necessary for fluent natural language generation. However, as with all research, we have found many more questions than answers: there are plenty of leftovers waiting to be consumed.

References

Akmajian, A. and Lehrer, A. [1976] NP-like Quantifiers and the Problem of Determining the Head of an NP. *Linguistic Inquiry*, **2**.

Allen, J. F. and Perrault, C. R. [1978] Participating in Dialogues: Understanding via Plan Deduction. In *Proceedings of the Second National Conference of the Canadian Society for the Study of Computational Intelligence*, Toronto, Canada, 1978, pp214–223.

Allen, J. F. and Perrault, C. R. [1980] Analyzing Intention in Dialogues. *Artificial Intelligence*, **15**, 143–178.

Allen, J. F. [1983] Recognizing intentions from natural language utterances. Chapter 2 in Brady, M. and Berwick, R. C. (eds.) *Computational Models of Discourse*, pp107–164. Cambridge, Mass.: MIT Press.

Alshawi, H. [1987] *Memory and Context for Language Interpretation.* Cambridge: Cambridge University Press.

Appelt, D. E. [1982] Planning Natural-Language Utterances to Satisfy Multiple Goals. Technical Note No. 259, SRI International, Menlo Park, Ca., March, 1982.

Appelt, D. E. [1986] *Planning English Sentences.* Cambridge: Cambridge University Press.

Austin, J. L. [1962] *How To Do Things With Words.* Oxford: Clarendon Press.

Bach, E. [1986] The algebra of events. *Linguistics and Philosophy*, **9**, 5–16.

Baker, C. L. [1978] *Introduction to Generative-Transformational Syntax.* Englewood Cliffs, N.J.: Prentice-Hall.

Bartsch, R. [1975] Subcategorization of adnominal and adverbial modifiers. In Keenan, E. L. (ed.) *Formal Semantics of Natural Language: Papers from a colloquium sponsored by King's College Research Centre, Cambridge*, pp175–187. Cambridge: Cambridge University Press.

Bernardo, R. [1977] The Cognitive Relevance of the Sentence. Masters Thesis, University of California at Berkeley.

Bobrow, D. and Winograd, T. [1977] An overview of KRL-0, a knowledge representation language. *Cognitive Science*, **1**.

Bosch, P. [1983] *Agreement and anaphora.* London: Academic Press.

Brachman, R. J. [1979] Taxonomy, Descriptions, and Individuals in Natural Language Understanding. In *Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics*, University of California at San Diego, La Jolla, Ca., August, 1979, pp33–38.

Brachman, R. J. and Levesque, H. J. (eds.) [1985] *Readings in Knowledge Representation.* Los Altos, Ca.: Morgan Kaufmann Publishers, Inc.

Bree, D. S. and Smit, R. A. [1986] Linking Propositions. In *Proceedings of the 11th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Institut fuer Kommunikationsforschung und Phonetik, Bonn University, Bonn, August, 1986, pp177–180.

Bruce, B. C. [1975a] Belief Systems and Language Understanding. BBN Report No. 2973, Bolt, Beranek and Newman Inc., Cambridge, Mass., January, 1975.

Bruce, B. C. [1975b] Case Systems for Natural Language. *Artificial Intelligence*, **6**, 327–360.

Busemann, S. [1984] Topicalization and Pronominalization: Extending a Natural Language Generation System. Report ANS-28, University of Hamburg, Hamburg, June, 1984.

Butterworth, B. [1975] Hesitation and Semantic Planning in Speech. *Journal of Psycholinguistic Research*, **4**, 75–87.

Calder, J., Klein, E. and Zeevat, H. [1988] Unification Categorial Grammar: A Concise, Extendable Grammar for Natural Language Processing. In *Proceedings of the 12th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Budapest, August, 1988.

Carlson, L. [1981] Aspect and quantification. In Tedeschi, P. and Zaenen, A. (eds.) *Syntax and Semantics*, Volume 14: *Tense and Aspect*, pp31–64. New York: Academic Press.

Carter, D. [1987] *Interpreting Anaphors in Natural Language Texts.* Chichester: Ellis Horwood.

Chafe, W. L. [1977] Creativity in Verbalization and its Implications for the Nature of Stored Knowledge. In Freedle, R. O. (ed.) *Discourse Production and Comprehension*, Volume 1, pp41–55. Norwood, N.J.: Ablex.

Chafe, W. L. [1979] The Flow of Thought and the Flow of Language. In Givon, T. (ed.) *Syntax and Semantics*, Volume 12: *Discourse and Syntax.* New York: Academic Press.

Charniak, E. [1976] On the referential/attributive distinction. Working Paper No. 24, ISSCO Institut Dalle Molle, Geneva, Switzerland, 1976.

Charniak, E. and McDermott, D. [1985] *Introduction to Artificial Intelligence.*

Reading, Mass.: Addison-Wesley.

Clark, H. H. and Marshall, C. R. [1981] Definite Reference and Mutual Knowledge. In Joshi, A. K., Webber, B. L. and Sag, I. A. (eds.) *Elements of Discourse Understanding*, pp10–63. Cambridge: Cambridge University Press.

Clocksin, W. F. and Mellish, C. S. [1981] *Programming in Prolog.* Berlin: Springer-Verlag.

Cohen, P. R. [1978] On Knowing What to Say: Planning Speech Acts. Technical Report No. 118, University of Toronto, Toronto, 1978.

Cohen, P. R. and Perrault, C. R. [1979] Elements of a Plan-Based Theory of Speech Acts. *Cognitive Science*, **3**, 177–212.

Cohen, P. R. and Levesque, H. J. [1980] Speech Acts and the Recognition of Shared Plans. In *Proceedings of the Third Conference of the Canadian Society for Computational Studies of Intelligence*, Victoria, B.C., Canada, 1980, pp263–271.

Cohen, P. R. [1981] The Need for Identification as a Planned Action. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, University of British Columbia, Vancouver, B.C., Canada, August 24–28, 1981.

Cohen, P. R. [1984a] The Pragmatics of Referring and the Modality of Communication. *American Journal of Computational Linguistics*, **10**, 97–146.

Cohen, P. R. [1984b] Referring as Requesting. In *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, Ca., 2–6 July, 1984, pp207–211.

Cohen, R. [1984] A Computational Theory of the Function of Clue Words in Argument Understanding. In *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, Ca., 2–6 July, 1984, pp251–258.

Crystal, D. [1985] *A Dictionary of Linguistics and Phonetics*, 2nd Edition. Oxford: Basil Blackwell.

Danks, J. H. [1977] Producing Ideas and Sentences. In Rosenberg, S. (ed.) *Sentence Production: Development in Research and Theory.* Hillsdale, N.J.: Lawrence Erlbaum Associates.

Danlos, L. [1984] Conceptual and Linguistic Decisions in Generation. In *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, Ca., 2–6 July, 1984, pp501–504.

Danlos, L. [1987] *The Linguistic Basis of Text Generation.* Cambridge: Cambridge University Press.

Davey, A. [1978] *Discourse Production.* Edinburgh: Edinburgh University Press.

Davidson, D. [1967] Truth and meaning. *Synthese*, **17**, 304–323.

Donnellan, K. S. [1966] Reference and Definite Descriptions. *Philosophical Review*, **75**, 281–304.

Eisele, A. and Dorre, J. [1986] A Lexical Functional Grammar System in Prolog. In *Proceedings of the 11th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Institut fuer Kommunikationsforschung und Phonetik, Bonn University, Bonn, August, 1986, pp551–553.

Fauconnier, G. [1985] *Mental Spaces: Aspects of Meaning Construction in Natural Language*. Cambridge, Mass.: MIT Press.

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

Fillmore, C. [1968] The Case for Case. In Bach, E. and Harms, R. T. (eds.) *Universals in Linguistic Theory*. New York: Holt, Rinehart and Winston.

Frege, G. [1892] On Sense and Meaning. *Zeitschrift fur Philosophie und philosophische Kritik*, **100**, 25–50. Translation by Max Black in Frege (1984) pp157–177.

Gazdar, G., Klein, E., Pullum, G. and Sag, I. [1985] *Generalized Phrase Structure Grammar*. London: Basil Blackwell.

Goodman, B. A. [1985] Repairing Reference Identification Failures by Relaxation. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois, July, 1985, pp204–217.

Goodman, B. A. [1986] Reference Identification and Reference Identification Failures. *Computational Linguistics*, **12**, 273–305.

Granville, R. [1984] Controlling Lexical Substitution in Computer Text Generation. In *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, Ca., 2–6 July, 1984, pp381–384.

Grice, H. P. [1975] Logic and Conversation. In Cole, P. and Morgan, J. L. (eds.) *Syntax and Semantics*, Volume 3: *Speech Acts*, pp41–58. New York: Academic Press.

Grimes, J. [1978] Topic Levels. In Waltz, D. L. (ed.) *Theoretical Issues in Natural Language Processing-2*, University of Illinois at Urbana-Champaign, Urbana, Illinois, July 25–27, 1978.

Grimes, J. E. [1982] Reference Spaces in Text. In Allen, S. (ed.) *Proceedings of the 51st Nobel Symposium*, Stockholm, Sweden, 1982, pp381–413.

Grosz, B. J. [1977] The Representation and Use of Focus in Dialogue. Technical Note No. 151, SRI International, Menlo Park, Ca., July, 1977.

Grosz, B. J. [1978] Discourse Knowledge. In Walker, D. E. (ed.) *Understanding*

Spoken Language, pp229–344. New York: North Holland.

Grosz, B. J. and Sag, I. A. [1981] Focusing and Description in Natural Language Dialogs. In Joshi, A. K. and Webber, B. L. (eds.) *Elements of Discourse Understanding*, pp84–105. Cambridge: Cambridge University Press.

Grosz, B. J., Joshi, A. K. and Weinstein, S. [1983] Providing a Unified Account of Definite Noun Phrases in Discourse. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Massachusetts Institute of Technology, Cambridge, Mass., 15–17 June, 1983, pp44–49.

Grosz, B. J. and Sidner, C. L. [1985] Discourse Structure and the Proper Treatment of Interruptions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, University of California at Los Angeles, Los Angeles, Ca., August 18–23, 1985, pp832–839.

Grosz, B. J. and Sidner, C. L. [1986] Attention, Intentions, and the Structure of Discourse. *Computational Linguistics*, **12**, 175–204.

Guindon, R. [1985] Anaphora Resolution: Short-Term Memory and Focusing. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois, 8–12 July, 1985, pp218–227.

Halliday, M. A. K. [1973] *Explorations in the Functions of Language.* London: Edward Arnold.

Halliday, M. A. K. and Hasan, R. [1976] *Cohesion in English.* London: Longman.

Halliday, M. A. K. [1985] *An Introduction to Functional Grammar.* London: Edward Arnold.

Hammond, K. J. [1986] CHEF: A Model of Case-based Planning. In *Proceedings of the 5th Annual Meeting of the American Association for Artificial Intelligence*, Philadelphia, Pa., 1986, pp267–271.

Hankamer, J. and Sag, I. [1976] Deep and Surface Anaphora. *Linguistic Inquiry*, **7**, 391–426.

Hawkins, J. A. [1978] *Definiteness and Indefiniteness.* London: Croom Helm.

Hayes, P. J. [1974] Some Problems and Non-Problems in Representation Theory. In *Proceedings of the* AISB *Summer Conference*, University of Sussex, 1974, pp63–79. Reprinted in Brachman and Levesque [1985], pp3–23.

Hayes, P. J. [1978] The Naive Physics Manifesto. In Michie, D. and Meltzer, B. (eds.) *Machine Intelligence 4.* Edinburgh: Edinburgh University Press.

Hayes, P. J. [1985] The Second Naive Physics Manifesto. In Brachman, R. J. and Levesque, H. J. (eds.) *Readings in Knowledge Representation*, pp468–485. Los Altos, Ca.: Morgan Kaufmann Publishers, Inc.

Hellwig, P. [1986] Dependency Unification Grammar. In *Proceedings of the 11th*

International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics, Institut fuer Kommunikationsforschung und Phonetik, Bonn University, Bonn, August, 1986, pp195–198.

Herrmann, T. and Laucht, M. [1976] On Multiple Verbal Codability of Objects. *Psychological Research*, **38**, 355–368.

Hirst, G. [1981a] *Anaphora in Natural Language Understanding.* Berlin: Springer-Verlag.

Hirst, G. [1981b] Discourse-Oriented Anaphora Resolution: A Review. *American Journal of Computational Linguistics*, **7**, 85–98.

Hobbs, J. R. [1978] Resolving Pronoun References. *Lingua*, **44**, 311–338.

Hobbs, J. H. and Evans, D. A. [1979] Conversation as Planned Behavior. Technical Note No. 203, SRI International, Menlo Park, Ca., December, 1979.

Hobbs, J. R. [1985] Ontological Promiscuity. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois, July, 1985, pp61–69.

Hornstein, N. and Lightfoot, D. (eds.) [1981] *Explanation in Linguistics: The logical problem of language acquisition.* London: Longman.

Houghton, G. [1986] The Production of Language in Discourse: A Computational Model. PhD Thesis, University of Sussex.

Jameson, A. and Wahlster, W. [1982] User modelling in anaphora generation: ellipsis and definite description. In ECAI82, 1982, pp222–227.

Johnson-Laird, P. N. and Garnham, A. [1980] Descriptions and discourse models. *Linguistics and Philosophy*, **3**, 371–393.

Johnson-Laird, P. N. [1983] *Mental Models.* Cambridge: Cambridge University Press.

Joshi, A. [1983] Factoring recursion and dependencies: an aspect of tree-adjoining grammars (TAG) and a comparison of some formal properties of TAGs, GPSGs, PLGs, and LFGs. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Massachusetts Institute of Technology, Cambridge, Mass., 15–17 June, 1983, pp7–15.

Joshi, A. K. and Vijay-Shankar, K. [1985] Some Computational Properties of Tree Adjoining Grammars. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois, July, 1985, pp82–93.

Kamp, J. A. W. [1975] Two Theories about Adjectives. In Keenan, E. L. (ed.) *Formal Semantics of Natural Language: Papers from a colloquium sponsored by King's College Research Centre, Cambridge*, pp123–155. Cambridge: Cambridge University Press.

Kamp, H. [1981] A theory of truth and semantic representation. In Groenendijk, J. A. G., Janssen, T. M. V. and Stokhof, M. B. J. (eds.) *Formal Methods in the Study of Language*, Volume 136, pp277–322. Amsterdam: Mathematical Centre Tracts.

Kantor, R. N. [1977] The Management and Comprehension of Discourse Connection by Pronouns in English. PhD Thesis, The Ohio State University.

Karlin, R. F. [1988] Defining the Semantics of Verbal Modifiers in the Domain of Cooking Tasks. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, State University of New York at Buffalo, Buffalo, N.Y., 7–10 June, 1988, pp61–67.

Karttunen, L. [1976] Discourse referents. In McCawley, J. (ed.) *Syntax and Semantics*, Volume 7, pp363–386. New York: Academic Press.

Karttunen, L. [1986] D-PATR: A Development Environment for Unification-Based Grammars. In *Proceedings of the 11th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Institut fuer Kommunikationsforschung und Phonetik, Bonn University, Bonn, 25–29 August, 1986, pp74–80.

Kay, M. [1975] Syntactic Processing and Functional Sentence Perspective. In Schank, R. and Nash-Webber, B. L. (eds.) *Theoretical Issues in Natural Language Processing*, Cambridge, Mass, June 10–13, 1975.

Keenan, E. L. and Faltz, L. M. [1978] *Logical Types for Natural Language.* UCLA Occasional Papers in Linguistics, No. 3.

Kempen, G. and Hoenkamp, E. [1982] An Incremental Procedural Grammar for Sentence Formulation. Internal Report No. 82 FU 14, Katholieke Universiteit Nijmegen, The Netherlands, June, 1982.

Kronfeld, A. [1986] Donnellan's Distinction and a Computational Model of Reference. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, Columbia University, New York, N.Y., June, 1986, pp186–191.

Kronfeld, A. [1988] Donnellan's Distinction as an Adequacy Test for a Referring Model. Technical Note No. 436, SRI International, Menlo Park, Ca., April, 1988.

Kucera, H. and Francis, W. N. [1967] *Computational Analysis of Present-Day American English.* Providence, Rhode Island: Brown University Press.

Lasnik, H. [1976] Remarks on Coreference. *Linguistic Analysis*, **2**, 1–22.

Leech, G. and Svartvik, J. [1975] *A Communicative Grammar of English.* London: Longman.

Lehrer, A. [1969] Semantic cuisine. *Journal of Linguistics*, **5**, 39–55.

Lehrer, A. [1972] Cooking vocabularies and the culinary triangle of Levi-Strauss. *Anthropological Linguistics*, **14**, 155–171.

Lewis, D. [1983] Scorekeeping in a Language Game. Chapter 13 in *Philosophical Papers*, Volume 1. Oxford: Oxford University Press.

Linde, C. [1979] Focus of Attention and the Choice of Pronouns in Discourse. In Givon, T. (ed.) *Syntax and Semantics*, Volume 12: *Discourse and Syntax*. New York: Academic Press.

Lindsey, J. R. [1975] Producing Simple Utterances: How Far Ahead Do We Plan? *Cognitive Psychology*, **7**, 1–19.

Lindsey, J. R. [1976] Producing Simple Utterances: Details of the Planning Process. *Journal of Psycholinguistic Research*, **5**, 331–354.

Link, G. [1983] The logical analysis of plurals and mass terms: a lattice-theoretical approach. In Bauerle, R., Schwarze, C. and von Stechow, A. (eds.) *Meaning, Use and Interpretation of Language*, pp302–323. Berlin: de Gruyter.

Lyons, J. [1968] *Introduction to Theoretical Linguistics*. Cambridge: Cambridge University Press.

Lyons, J. [1977] *Semantics*. Cambridge: Cambridge University Press.

MacNeilage, P. [1973] Linguistic Units and Speech Production. Presented at the 85th Meeting of the Acoustical Society of America, Boston, Mass., April 13, 1973.

Mann, W., Bates, M., Grosz, B. J., McDonald, D. D., McKeown, K. R. and Swartout, W. [1982] Text Generation. *American Journal of Computational Linguistics*, **8**, 62–69.

Mann, W. C. and Moore, J. [1982] Computer generation of multiparagraph English text. *American Journal of Computational Linguistics*, **7**, 17–29.

Mann, W. C. [1983] An overview of the nigel text generation grammar. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Massachusetts Institute of Technology, Cambridge, Mass., 15–17 June, 1983, pp79–84.

Matthiessen, C. M. I. M. [1981] A Grammar and a Lexicon for a Text-Production System. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, Ca., June/July, 1981, pp49–55.

Matthiessen, C. M. I. M. [1984] How to Make Grammatical Choices in Text Generation. Report RS-83-120, USC Information Sciences Institute, Marina Del Rey, Ca., February, 1984. Reprinted from The Tenth LACUS Forum 1983.

Matthiessen, C. and Kasper, R. [1987] Systemic Grammar and Functional Unification Grammar and Representational Issues in Systemic Functional Grammar. Technical Report No. RS-87-179, USC Information Sciences Institute, Marina Del Rey, Ca., April, 1987.

Mayo, B. [1961] Objects, Events, and Complementarity. *Philosophical Review*, **LXX**, 340–361.

McCoy, K. F. [1985] The Role of Perspective in Responding to Property Misconceptions. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, University of California at Los Angeles, Los Angeles, Ca., August 18–23, 1985, pp791–793.

McDonald, D. D. [1977] Linguistic Reasoning During Language Generation. Technical Report No. 404, MIT Artificial Intelligence Laboratory, 1977.

McDonald, D. D. [1978] Subsequent Reference: Syntactic and Rhetorical Constraints. In Waltz, D. L. (ed.) *Theoretical Issues in Natural Language Processing–2*, University of Illinois at Urbana-Champaign, Urbana, Illinois, July 25–27, 1978, pp64–72.

McDonald, D. D. [1979] Steps Towards a Psycholinguistic Model of Language Production. Working Paper No. 193, MIT Artificial Intelligence Laboratory, April, 1979.

McDonald, D. D. [1980a] Natural Language Generation as a Process of Decision-Making under Constraints. PhD Thesis, Department of Computer Science and Electrical Engineering, MIT.

McDonald, D. D. [1980b] A Linear-Time Model of Language Production: Some Psycholinguistic Implications. In *Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics and Parasession on Topics in Interactive Discourse*, University of Pennsylvania, Philadelphia, Pa., June 19–22, 1980, pp55–57.

McDonald, D. D. [1981] Language Production: The Source of the Dictionary. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, Ca., June 29 – July 1, 1981, pp57–61.

McDonald, D. D. [1981] MUMBLE, A Flexible System for Language Production. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, University of British Columbia, Vancouver, B.C., Canada, August 24–28, 1981, pp1062.

McDonald, D. D. [1983] Description Directed Control: Its Implications for Natural Language Generation. In Cercone, N. (ed.) *Computational Linguistics*. London: Pergamon Press.

McDonald, D. D. and Pustejovsky, J. D. [1985] TAGs as a Grammatical Formalism for Generation. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, Illinois, July, 1985, pp94–103.

McKeown, K. R. [1982] Generating Natural Language Text in Response to Questions about Database Structure. PhD Thesis, University of Pennsylvania.

McKeown, K. R. [1985] *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge: Cambridge University Press.

Mellish, C. S. [1987] Implementing Systemic Classification by Unification.

Presented at the workshop on Natural Language Processing, Unification and Grammatical Formalisms, Stirling, 10–12 June, 1987.

Mellish, C. [1988] Natural Language Generation from Plans. Chapter 7 in Zock, M. and Sabah, G. (eds.) *Advances in Natural Language Generation*, Volume 1: *An Interdisciplinary Perspective*, pp131–145. London: Pinter Publishers Ltd.

Moore, J. A. and Mann, W. C. [1979] A Snapshot of KDS, a Knowledge Delivery System. In *Proceedings of the 17th Annual Meeting of the Association for Computational Linguistics*, University of California at San Diego, La Jolla, Ca., August, 1979, pp51–52.

Ney, J. W. [1983] Optionality and choice in the selection of order of adjectives in English. *General Linguistics*, **23**, 94–128.

Novak, H. [1987] Generating Referring Phrases in a Dynamic Environment. In Zock, M. and Sabah, G. (eds.) *Proceedings of the First European Workshop on Language Generation*, Abbey de Royaumont, 23–26 January, 1987.

Nunberg, G. [1978] *The Pragmatics of Reference.* Bloomington, Indiana: Indiana University Linguistics Club.

Ochs, E. [1979] Planned and Unplanned Discourse. In Givon, T. (ed.) *Syntax and Semantics*, Volume 12: *Discourse and Syntax*, pp51–80. New York: Academic Press.

Ogden, C. K. and Richards, I. A. [1949] *The Meaning of Meaning*, 10th Edition. London: Routledge and Kegan Paul.

Partee, B. [1978] Bound Variables and Other Anaphors. In Waltz, D. L. and Waltz, D. (eds.) *Theoretical Issues in Natural Language Processing–2*, University of Illinois at Urbana-Champaign, Urbana, Illinois, July 25–27, 1978, pp79–85.

Pelletier, F. J. (ed.) [1979] *Mass Terms: Some Philosophical Problems.* Dordrecht: D. Reidel.

Pinkal, M. [1986] Definite Noun Phrases and the Semantics of Discourse. In *Proceedings of the 11th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Institut fuer Kommunikationsforschung und Phonetik, Bonn University, Bonn, 25–29 August, 1986, pp368–373.

Polanyi, L. and Scha, R. [1984] A Syntactic Approach to Discourse Semantics. In *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, Ca., 2–6 July, 1984, pp413–419.

Polanyi, L. [1985] A Theory of Discourse Structure and Discourse Coherence. In Eilfort, W. H., Kroeber, P. D. and Peterson, K. L. (eds.) *Papers from the General Session at the Twenty-First Regional Meeting of the Chicago Linguistics Society*, Chicago, April 25–27, 1985.

Pollack, J. and Waltz, D. [1985] Massively Parallel Parsing: A Strongly

Interactive Model of Natural Language Interpretation. *Cognitive Science*, No. 9, 51–74.

Popowich, F. P. [1988] Reflexives and Tree Unification Grammar. PhD Thesis, Centre for Cognitive Science, University of Edinburgh.

Prince, E. [1981] A Taxonomy of Given-New Information. In Cole, P. (ed.) *Radical Pragmatics.* New York: Academic Press.

Quine, W. V. O. [1960] *Word and Object.* Cambridge, Mass.: MIT Press.

Quirk, R., Greenbaum, S., Leech, G. and Svartvik, J. [1985] *A Comprehensive Grammar of the English Language.* London: Longman.

Reddy, M. J. [1979] The Conduit Metaphor: A Case of Frame Conflict in Our Language about Language. In Ortony, A. (ed.) *Metaphor and Thought.* Cambridge: Cambridge University Press.

Reichenbach, H. [1947] *Elements of Symbolic Logic.* London: Macmillan.

Reichgelt, H. [1986] Reference and Quantification in the Cognitive View of Language. PhD Thesis, School of Epistemics, University of Edinburgh.

Reichman, R. [1978] Conversational Coherency. *Cognitive Science*, **2**, 283–327.

Reichman, R. [1981] Plain Speaking: A Theory and Grammar of Spontaneous Discourse. PhD Thesis. Published as BBN Report No. 4681.

Reichman, R. [1985] *Getting Computers to Talk Like You and Me.* Cambridge, Mass.: MIT Press.

Reinhart, T. [1976] The Syntactic Domain of Anaphora. PhD Thesis, Department of Foreign Literature and Linguistics, MIT.

Reinhart, T. [1981] Definite NP anaphora and C-Command Domains. *Linguistic Inquiry*, **12**, 605–635.

Reinhart, T. [1983] Coreference and bound anaphora: A restatement of the anaphora questions. *Linguistics and Philosophy*, **6**, 47–88.

Reithinger, N. [1986] Generating Referring Expressions and Pointing Gestures. XTRA Report No. 13, Universitat des Saarlandes, November, 1986.

Ritchie, G. [1985] The Referring Expression Algorithm from Davey [1974]: An Approximate Reconstruction. Unpublished paper.

Rosch, E., Mervis, C., Gray, W., Johnson, D. and Boyes-Braem, P. [1976] Basic Objects in Natural Categories. *Cognitive Psychology*, **8**, 382–439.

Rubinoff, R. [1986] Adapting Mumble: Experience with Natural Language Generation. Technical Report No. MS-CIS-86-32, Department of Computer and Information Science, University of Pennsylvania.

Russell, B. [1902] On Denoting. *Mind*, **14**, 479–493.

Russell, B. [1919] *Introduction to Mathematical Philosophy.* London: George Allen and Unwin.

Sacerdoti, E. D. [1974] Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, **5**, 115–135.

Sacerdoti, E. D. [1977] *A Structure for Plans and Behavior.* New York: North Holland.

Sanford, A. J. and Garrod, S. C. [1981] *Understanding Written Language.* Chichester: John Wiley and Sons.

Searle, J. [1969] *Speech Acts: An Essay in the Philosophy of Language.* Cambridge: Cambridge University Press.

Selkirk, E. O. [1977] Some Remarks on Noun Phrase Structure. In Culicover, P. W., Wasow, T. and Akmajian, A. (eds.) *Formal Syntax.* New York: Academic Press.

Shieber, S. M. [1986] *An Introduction to Unification-based Approaches to Grammar.* Chicago, Illinois: The University of Chicago Press.

Shieber, S. M., Pereira, F. C. N., Karttunen, L. and Kay, M. [1986] A Compilation of Papers on Unification-Based Grammar Formalisms Parts I and II. Report No. CSLI-86-48, Center for the Study of Language and Information, April, 1986.

Shieber, S. M. [1988] A Uniform Architecture for Parsing and Generation. Technical Note No. 437, SRI International, Menlo Park, Ca., May 2, 1988.

Sidner, C. L. [1979] Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse. Technical Report No. 537, MIT Artificial Intelligence Laboratory, June, 1979.

Sidner, C. L. [1981] Focusing for Interpretation of Pronouns. *American Journal of Computational Linguistics*, **7**, 217–231.

Slocum, J. [1975] Speech Generation from Semantic Nets. Technical Note No. 115, Stanford Research Institute Artificial Intelligence Center, September, 1975. Presented at the Thirteenth Annual Meeting of the Association for Computational Linguistics, Boston, Massachusetts, 30 October–1 November 1975.

Slocum, J. [1978] Generating a Verbal Response. Chapter XVII in Walker, D. E. (ed.) *Understanding Spoken Language*, pp375–380. New York: North Holland.

Strawson, P. [1950] On Referring. *Mind*, **59**, 320–44.

Swartout, W. R. [1983] XPLAIN: A System for Creating and Explaining Expert Consulting Programs. ISI Reprint Series No. RS-83-4, USC Information Sciences Institute, Marina Del Rey, Ca., July, 1983.

Tate, A. [1976] Project Planning Using a Hierarchic Non-Linear Planner. Research Report No. 245, Department of Artificial Intelligence, University of Edinburgh, 1976.

Tate, A. [1985] A Review of Knowledge-based Planning Techniques. *Knowledge*

Engineering Review, No. 2, 4–16.

Taylor, I. [1969] Content and Structure in Sentence Production. *Journal of Verbal Learning and Verbal Behavior*, **8**, 170–175.

Thompson, H. [1977] Strategy and Tactics in Language Production. In Beach, W. A., Fox, S. E. and Philosoph, S. (eds.) *Papers from the Thirteenth Regional Meeting of the Chicago Linguistics Society*, Chicago, April 14–16, 1977.

Tsang, E. P. K. [1986] Plan Generation in a Temporal Frame. In *Proceedings of the Seventh European Conference on Artificial Intelligence*, Brighton, UK, 21–25 July, 1986, pp479–493.

Uszkoreit, H. [1986] Categorial Unification Grammars. In *Proceedings of the 11th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, Institut fuer Kommunikationsforschung und Phonetik, Bonn University, Bonn, 25–29 August, 1986, pp187–194.

Webber, B. L. [1979] *A Formal Approach to Discourse Anaphora.* London: Garland Publishing.

Webber, B. L. [1983] So what can we talk about now? Chapter 6 in Berwick, R. C. and Brady, M. (eds.) *Computational Models of Discourse*, pp331–371. Cambridge, Mass.: MIT Press.

Webber, B. L. [1988] Discourse Deixis: Reference to Discourse Segments. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, State University of New York at Buffalo, Buffalo, N.Y., 7–10 June, 1988, pp113–122.

Wilks, Y. [1975] Preference Semantics. In Keenan, E. L. (ed.) *The Formal Semantics of Natural Language*, pp329–350. Cambridge: Cambridge University Press.

Winograd, T. [1972] *Understanding Natural Language.* New York: Academic Press.

Wykes, T. [1981] Inference and Children's Comprehension of Pronouns. *Journal of Experimental Child Psychology*, **32**, 264–278.

Zeevat, H., Klein, E. and Calder, J. [1987] An Introduction to Unification Categorial Grammar. In Haddock, N. J., Klein, E. and Morrill, G. (eds.) *Edinburgh Working Papers in Cognitive Science*, Volume 1: *Categorial Grammar, Unification Grammar, and Parsing.*

Zemach, E. [1979] Four ontologies. In Pelletier, F. J. (ed.) *Mass Terms: Some Philosophical Problems*, pp63–80.