

From Data to Speech:
Language Generation in Context

The work described in this thesis has been carried out at IPO, Center for User-System Interaction, Eindhoven, within the framework of the Priority Programme Language and Speech Technology (TST), which is sponsored by NWO (Netherlands Organisation for Scientific Research).

© M. Theune, 2000.

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Theune, Mariët.

From data to speech: language generation in context /
by Mariët Theune. –

Eindhoven: Eindhoven University of Technology, 2000. - Thesis

ISBN 90-386-0833-0

NUGI 832

Keywords: Language generation

Printed by: Universiteitsdrukkerij, Eindhoven University of Technology

From Data to Speech:
Language Generation in Context

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr. M. Rem, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op vrijdag 1 december 2000 om 16.00 uur

door

Mariët Theune

geboren te Vlissingen

Dit proefschrift is goedgekeurd door de promotoren:

prof.ir. S.P.J. Landsbergen

en

prof.dr. R.P.G. Collier

Copromotor:

dr. E.J. Kraemer

Acknowledgements

Many people have helped me in some way or other during my PhD-research. Those who made the most direct contributions to my work are mentioned below. I wish to thank them, as well as all the others who are not mentioned.

Of all the people who contributed to my research, Jan Landsbergen and Emiel Kraemer certainly deserve to be mentioned first. Their advice and support have been invaluable to me. They read many versions of this thesis (and of practically everything else that I wrote), and provided me with countless useful, detailed suggestions for improvement.

Jan forced me, friendly but insistent, to perfect my algorithms and to clarify everything that I had left conveniently vague. Having stood at the origin of my PhD-project, Jan kept watching over it until the end, first staying on at IPO just for me and then continuing to ‘coach’ me after his retirement. I could not have wished for a better supervisor.

The same holds for Emiel, my daily supervisor. Ever cheerful and positive, Emiel always made me look on the bright side of things. We have cooperated closely, most notably on the generation of referring expressions. Emiel’s continuing stream of good ideas and infectious enthusiasm made it a pleasure to work together.

This thesis has benefited from several useful suggestions by René Collier and Jacques Terken. Jacques also provided helpful comments on several of my earlier publications. Here, I should also mention the other members of my thesis committee: Remko Scha and Paul de Bra; and last but not least, Stephen Pulman and Donia Scott, who agreed to come over all the way from England, just for my thesis defence.

During the first year-and-a-half of my PhD-research, Jan Odijk acted as my daily supervisor. I am grateful to him for giving me a good start. During my PhD-research, my path kept crossing that of Kees van Deemter, with whom I shared several interests. Among other things, I

am obliged to Kees for commenting on several papers and on Chapter 4 of this thesis. Jan, Kees and the others who were involved in the DYD-project provided me with the framework for my research.

Esther Klabbers put up with *me* as a room mate for nearly five years. (Note the contrastive accent!) I think we made a good team. In addition, Esther contributed to this thesis by providing me with material for the sections on speech generation, and by reading the final manuscript to correct my English.

Danny Kersten patiently suffered being treated as my personal help desk, never complaining when I came to him with all my computer-related problems. He developed the GoalGetter Internet-demo, which provided Esther and me with a great way to illustrate our research. (Although I must admit that we sometimes got fed up with giving all those demonstrations.)

With Marc Swerts and Mieke Weegels, my fellow M's, I have had a very pleasant cooperation (together with Emiel). We had a lot of fun and even managed to get some work done.

I am also indebted to everybody who participated in the experiments described in this thesis, and to John-Pierre Verhagen and Daniël Nachtegaal, who conducted two of these experiments.

It would be too much to mention everybody individually, so I will just say thanks in general to all other, former and current, IPO-colleagues from whom I have learned a lot and who made me feel at home at IPO; in particular the people from SLI. (Tea breaks *are* relevant!)

Finally, I wish to mention my family and friends, who provided me with a life outside work and showed a continued interest in my progress.

Contents

1	Introduction	1
1.1	Language generation in context	1
1.2	Text generation	2
1.2.1	Text generation tasks	3
1.2.2	Text generation systems	5
1.2.3	The generation of referring expressions	6
1.3	Language generation in data-to-speech	8
1.3.1	Speech generation methods	8
1.3.2	The importance of prosody	9
1.3.3	Prosody computation: an additional task	11
1.4	Research aims	12
1.5	Overview	13
2	Description of D2S	16
2.1	Introduction	16
2.2	The architecture of D2S	18
2.3	The GoalGetter system	19
2.4	Language generation in D2S	22
2.4.1	Architecture	22
2.4.2	Syntactic templates	25
2.4.3	Topics, conditions and coherence	29
2.4.4	Algorithm	31
2.4.5	Example	34
2.5	Prosody computation in D2S	39
2.5.1	Overview	40
2.5.2	Focus and information status	40
2.5.3	Weak and strong nodes	43
2.5.4	Phrase boundaries	44
2.5.5	Evaluation	45

2.6	Speech generation in D2S	46
2.6.1	Phonetics-to-speech	46
2.6.2	Phrase concatenation using prosodic variants	47
2.7	Discussion	49
3	Contrastive accent	52
3.1	Introduction	52
3.2	The need for contrastive accent	53
3.3	Early views on contrast	58
3.4	Formal approaches to contrastive accent assignment	61
3.4.1	Prevost: sets of alternatives	61
3.4.2	Pulman and others: HOU and parallelism	64
3.4.3	Van Deemter: contrariety and equivalence	67
3.4.4	Discussion	69
3.5	Experiment	71
3.5.1	Hypotheses and assumptions	71
3.5.2	Method	73
3.5.3	Materials	74
3.5.4	Results	76
3.5.5	Discussion	77
3.5.6	Coherence	78
3.5.7	Conclusions	80
3.6	Assigning contrastive accent within the LGM	81
3.6.1	Deriving contrast from data structures	81
3.6.2	Contrastive accent assignment: illustrations	84
3.6.3	A possible extension	90
3.6.4	Discussion	91
3.7	Summary	94
4	Generating descriptions in context	96
4.1	Introduction	96
4.2	The Incremental Algorithm (Dale and Reiter 1995)	98
4.2.1	Assumptions about domains	99
4.2.2	Outline of the Incremental Algorithm	100
4.2.3	Example	102
4.2.4	Discussion	102
4.3	A modification of the algorithm based on salience	104
4.3.1	Motivation: determining the context set	104
4.3.2	Definite descriptions and salience	107
4.3.3	Outline of the modified algorithm	108
4.3.4	Examples	112
4.3.5	Discussion	116

4.4	Determining salience weights	116
4.4.1	Hajičová: hierarchical focus constraints	117
4.4.2	Grosz et al.: Centering	118
4.4.3	Examples, predictions and comparison	119
4.4.4	Discussion: revising salience weight assignment	121
4.5	Further extensions	123
4.5.1	Pronominalisation	123
4.5.2	Relational descriptions	125
4.5.3	Bridging descriptions	139
4.6	Experiment	140
4.6.1	Hypotheses and assumptions	140
4.6.2	Method	142
4.6.3	Materials	142
4.6.4	Results	142
4.6.5	Discussion	143
4.7	Integration of the modified algorithm in the LGM	147
4.7.1	Generation of referring expressions	147
4.7.2	Referring expressions and accentuation	148
4.8	Summary	150
5	Language generation in a spoken dialogue system	152
5.1	Introduction	152
5.2	Using the LGM in OVIS	154
5.2.1	The OVIS architecture	154
5.2.2	Input and output of the LGM	156
5.2.3	Syntactic templates	159
5.2.4	Coherence	162
5.2.5	Variation	165
5.2.6	Discussion	166
5.3	Context and information status in dialogues	167
5.3.1	Grounding	168
5.3.2	Cross-speaker anaphora	169
5.3.3	Accentuation	172
5.3.4	Discussion	175
5.4	Detecting information status	177
5.4.1	Corpus	178
5.4.2	Linguistic cues	178
5.4.3	Prosodic cues	180
5.4.4	Discussion	183
5.5	Summary	183
6	Conclusion	185

x / CONTENTS

6.1	General overview	185
6.2	Future work	188
6.3	Final word	189
	Bibliography	191
	Samenvatting	205
	Curriculum Vitae	209

Introduction

1.1 Language generation in context

Natural language generation is the process of automatically creating a natural language text on the basis of a non-linguistic information representation, for instance information from a database. This thesis discusses the generation of texts that are to be conveyed to the user in spoken, not written, form. The fact that the generated texts must be pronounced places additional demands on language generation: the output should not be plain text, but so-called ‘enriched text’ containing markers that indicate the desired prosody of the spoken output.

As the subtitle of this thesis indicates, the research presented here focuses on language generation *in context*. The word ‘context’ allows for many different interpretations; those that are intended here are listed below.

Language generation and linguistic context. The focus of this thesis is on the relevance of linguistic context for generation. Here, linguistic context is informally defined as “what has been said previously in the text being generated”. In this thesis, it is shown how the coherence (and thus, naturalness and understandability) of generated texts can be increased by taking linguistic context into account during the generation of referring expressions and the computation of prosody (i.e., the assignment of prosodic markers to the generated output).

Language generation in a monologue or in a dialogue context. This thesis discusses the generation of spoken monologues, as well as language generation in spoken dialogue. The type of interaction between system and user, either monologue or dialogue, influences what counts as the linguistic context of an utterance. In monologue generation, the linguistic context consists only of the system output that preceded the

utterance currently being generated. In a dialogue, a distinction must be made between the user's and the system's contributions to the linguistic context. The fact that in a spoken dialogue the system can never be completely sure of what the user has said has consequences for prosody computation and other aspects of language generation.

Language generation in the context of data-to-speech. This thesis deals with language generation for data-to-speech systems, i.e., systems which produce spoken output. In the context of data-to-speech, an additional task of language generation is the placement of prosodic markers which indicate how the generated output should be pronounced. The more specific system context is formed by the D2S system.¹ This is a generic data-to-speech system developed at IPO, which has been the basis for various data-to-speech applications. The research presented here has resulted in several improvements to the language generation module of D2S.

The current chapter provides a brief overview of language and speech generation, forming the general background for the chapters that follow. Background information on more specific generation issues dealt with in this thesis is provided in the relevant chapters. The outline of the current chapter is as follows. Section 1.2 gives a global overview of *text generation*, i.e., language generation in systems that provide written output. The next section, Section 1.3, discusses the additional demands placed on language generation in systems that produce *spoken* output. In Section 1.4, the aims of the current research are presented. Finally, in Section 1.5 an overview of the thesis is given.

1.2 Text generation

Most research in the area of natural language generation has focused on the generation of texts that are presented to the user in written form. To distinguish this form of language generation from language generation aimed at spoken output, in this thesis the term *text generation* is used to denote the former. The more common term *natural language generation* is used for language generation in systems producing either written or spoken output.

In this section, a brief overview is given of the tasks involved in text generation (Section 1.2.1), and of the way these tasks are carried out in

¹D2S stands for 'data-to-speech'. This term describes D2S more appropriately than the more common 'concept-to-speech', as D2S takes as input data retrieved from tables or databases, rather than some sort of semantic or 'conceptual' representations.

current text generation systems (Section 1.2.2). Finally, an important generation task is highlighted: the generation of referring expressions (Section 1.2.3). For a comprehensive introduction to the field of natural language generation, the reader is referred to Reiter and Dale 2000.

1.2.1 Text generation tasks

Reiter and Dale 1997 distinguish six basic tasks that a text generation system should perform when going from its input data to a natural language text. These tasks can be summarised as follows:

1. *Content determination* is the task of converting the raw system input to the specific kind of data objects (or ‘messages’) that serve as a basis for the subsequent text creation processes. This includes deciding which input information to express in the text.
2. *Discourse planning* is the process of ordering the information to be expressed and determining the structure of the output text. The result is a text plan, often in the form of a tree structure representing the order and grouping of the messages, and the relations between them.
3. *Sentence aggregation* is the process of deciding which information to put in one sentence. Pieces of information that form separate input messages may be joined together and expressed using one sentence which, for instance, contains a conjunction or a relative clause.
4. *Lexicalisation* is the task of choosing the right words to express the input information. Often, a concept can be expressed using different words or phrases; the task of the lexicalisation procedure is to choose the one that is most appropriate in the given context.
5. *Referring expression generation* is the task of creating phrases to identify domain entities. This involves choosing the type of expression (e.g., a pronoun or a definite description) and in the case of definite descriptions, choosing the properties to include in the description.
6. *Linguistic realisation*, finally, is the task of creating grammatical sentences. This involves the application of syntactic and morphological rules that determine aspects like word order and agreement.

Although these tasks are listed more or less in the order of execution, some generation systems may perform them in a different order, as will be discussed below. It should also be noted that some authors (e.g., Cahill et al. 1999) have made somewhat different task distinctions than

Reiter and Dale 1997, for instance by leaving out or splitting up some of the tasks listed above. Still, the tasks presented here can be considered largely representative of the operations involved in text generation.

Reiter 1994 observes that most applied text generation systems make use of a pipeline architecture with the following three modules:

- Content determination
- Sentence planning
- Surface generation

In this architecture, the output of each module forms the input for its successor, and there is no feedback between modules. The modules can best be characterised in terms of their output: first, *content determination* produces an abstract text plan specifying the semantic structure of the text, then *sentence planning* maps the conceptual structures in the text plan onto linguistic sentence representations, and finally *surface realisation* converts these sentence plans to text. According to Reiter and Dale 1997, the generation tasks outlined in Section 1.2.1 are distributed over the different modules as follows: the *content determination* module, which Reiter and Dale call ‘text planner’, is responsible for the tasks of content determination and discourse planning; the *sentence planner* performs sentence aggregation, lexicalisation, and referring expression generation; and *surface generation* corresponds to linguistic realisation.

The relevance of the pipeline architecture is confirmed by Cahill et al. 1999, who discuss the architecture of 19 text generation systems, and show that most of these systems indeed conform to the pipeline architecture discussed in Reiter 1994. However, they also show that there is no standard mapping of tasks to modules in the pipeline. Some tasks, such as referring expression generation, can be performed in almost any module; other tasks seem to be on the boundary between two modules, and only a few tasks, such as discourse planning (which Cahill et al. divide into ‘ordering’ and ‘rhetorical structuring’) are generally associated with one specific module (in this case *content determination*).

The research by Cahill et al. 1999 shows that although existing systems diverge in their distribution of tasks over system modules, the pipeline architecture is the most common architecture. Still, several generation systems have a different architecture. As an example, Cahill et al. mention the KOMET system (Teich and Bateman 1994, Bateman and Teich 1995), which generates text through the traversal of a network.

Another example of a deviating architecture is the language generation module of IPO’s D2S system. As we will see in Chapter 2, the

language generation system employed in D2S performs text generation in a sentence-by-sentence fashion, without having subsequent modules that deal with different ‘levels’ of sentence production, as in the standard pipeline.

1.2.2 Text generation systems

In the ideal case, a text generation system should carry out each of the tasks discussed in Section 1.2.1 in a theoretically well-founded manner, for example by performing discourse planning on the basis of Rhetorical Structure Theory (Mann and Thompson 1985) and by using a grammar for linguistic realisation. In practice, however, such sophistication is usually reserved for only a few steps of the generation process. This limitation may have several causes. For instance, some systems are aimed (for research purposes) at only one particular generation task, e.g., linguistic realisation. Other tasks are then performed in a more *ad hoc* fashion (if they are performed at all). Another limiting factor for the deployment of linguistic rules in language generation is simply that not enough good linguistic rules are known yet (van Deemter et al. 1999). Finally, ‘linguistic’ generation techniques may lack computational speed and efficiency (see e.g., Bateman and Henschel 1999), which makes them less attractive for use in applied generation systems.

The alternative for ‘linguistic’ generation is the use of simple, *ad hoc* methods that are specific to particular applications. An example is the use of ‘canned text’, an approach which is based on pure string manipulation. As Reiter 1995 points out, linguistic notions hardly play any role in such ‘canned text’ approaches, which are mainly used for very simple applications in a limited domain. Building generation systems which use canned text is relatively quick and easy, and generation using canned text is both fast and efficient.

However, the use of such simple approaches has several disadvantages. First of all, output texts created using a ‘canned text’ approach tend to be rather simple (in particular at the discourse level) and show almost no variation.² In contrast, linguistic techniques allow for the generation of texts that are more complex and more coherent, making principled use of e.g., anaphora and rhetorical markers. Also, most canned text systems are entirely application-specific, and therefore not reusable. Linguistic methods are usually general in nature and therefore domain- and application independent. They are more flexible and easier to maintain. Generalising, we can say that canned text approaches offer

²Coch 1996 offers a discussion of the weak points of such texts, based on a formal evaluation of the quality of business reply letters, written by means of different techniques.

ease of development, computational speed and efficiency at the cost of text quality, generality and flexibility, whereas for linguistic generation techniques the opposite holds.

It is clear that for the generation of all but the most simple texts, at least some linguistic knowledge is required. However, given the current state-of-the-art it is hardly possible to build text generation systems where each generation task is fully guided by linguistic principles. As a consequence, most applied generation systems can be characterised as *hybrid* systems, in the sense that some generation tasks are carried out on the basis of linguistic notions, whereas other tasks are performed using a non-linguistic method, e.g., by using ready-made text strings. An example is the IDAS system (Reiter and Mellish 1993), which incorporates those portions of text that are difficult to generate linguistically as ‘canned text’ in the output. Other recent hybrid generation systems are described in Carenini et al. 1994, Geldof and van de Velde 1997, White and Caldwell 1998, and Busemann and Horacek 1998. The language generation module of D2S, the data-to-speech system developed at IPO, also employs a hybrid technique. For example, in D2S linguistic realisation is performed by means of hand-made sentence structures, but well-established rules are used for the generation of anaphoric expressions and the computation of prosody. To guide the performance of the latter tasks, the language generation module of D2S uses a model of the linguistic context. This model is used, among other things, to determine the type of referring expressions (e.g., pronouns or proper names). In Chapter 3, it is discussed how information on the linguistic context can also be used to determine the *content* of referring expressions. First, Section 1.2.3 gives an informal introduction to the generation of referring expressions.

1.2.3 The generation of referring expressions

The generation of expressions that identify particular entities is “one of the most ubiquitous tasks in language generation” (Dale and Reiter 1995: 233). It basically involves two kinds of decisions: determining the type of the expression, and selecting its content. Among the different types of expressions that can be used are proper names (e.g., *Bill Clinton*), definite descriptions (*the American president*) and pronouns (*he*). These types of expressions are subject to different restrictions on their use. For instance, generally a pronoun can only be used if there is an accessible antecedent for it in the linguistic context; it is not appropriate to use a pronoun when referring to a specific entity for the first time.³ In ad-

³In dialogues between humans, speakers sometimes use a pronoun for the initial reference to an entity which is highly prominent in the physical context. For instance,

dition, it is not appropriate to use a pronoun when there is more than one possible antecedent in the linguistic context, since this will result in an ambiguity. This is illustrated in (1) below. (Here, ‘??’ indicates that the continuation is not well-formed, in this case because the pronoun is ambiguous.)

- (1) Clinton and his friend entered the bar. ?? He ordered a drink.

Proper names and definite descriptions can be used for both initial and subsequent (i.e., anaphoric) references. However, in cases where there is only one possible antecedent in the linguistic context, a pronoun often seems most appropriate, as can be seen when comparing (2)a and (2)b.

- (2) a. Clinton entered the bar. Clinton ordered a drink
 b. Clinton entered the bar. He ordered a drink.

In general, it is considered ‘bad style’ to repeat the same description over and over in a text. More importantly, it may be confusing, since the impression might be created that the second name refers to a different entity from the first. It is more natural to start with a proper name or a full description (e.g., *the president of the United States*) and then to continue with a pronoun or a reduced description (e.g., *the president*).

This brings us to the issue of content selection, i.e., deciding which properties of the intended referent will be included in its description. In order to enable the hearer to identify the intended referent, those properties should be selected which help to distinguish it from other entities in the domain of discourse. For instance, when generating instructions for operating some device with many buttons, it would be useful to refer to a specific button as *the black button* if it is the only button of that colour. However, if there happen to be other black buttons on the device, it might be more appropriate to use a description like *the round button* (if all other buttons have some other shape) or even *the round black button* (if it is the only black button of that shape). So, which properties of an entity are selected for inclusion in its description, depends on the properties of the other entities with which it might be confused. Another relevant factor is the linguistic context: if an entity has just been mentioned, generally fewer properties are needed for the hearer to identify it. For instance, when the round black button of the previous

when a dog growls at a passer-by, its owner may say, “Don’t worry, it doesn’t bite.” Here, the dog has not been mentioned before, but there is no doubt of the pronoun’s intended referent. However, in applications of language generation such a ‘physical context’ is generally lacking.

example is mentioned for the second time, it can simply be referred to as *the button*, as shown in (3) below.

- (3) The₁ round black button is used for channel selection. To do this, turn the₁ button slowly to the left or right.

The influence of linguistic context on content selection is discussed in depth in Chapter 4.

1.3 Language generation in data-to-speech

The basic form of human communication is *spoken* language. Still, until recently research in language generation has been aimed mainly at the generation of *written* texts. At the same time, research in speech generation (usually called speech synthesis) has focused on the automatic production of human speech sounds, concentrating on acoustic/phonetic aspects without considering the preceding stages of speech production. Thus, for a long time text generation and speech generation have been regarded as separate fields, and they were rarely combined. However, recent years have seen a raised interest in systems with speech output. In some of these systems, the text that is to be pronounced is provided by a source outside the system. This is the case for spoken newspapers, email readers and other ‘text-to-speech’ applications. In other systems, such as spoken dialogue systems, the text to be pronounced is generated by the system itself. In these systems, language and speech generation are combined in the form of data-to-speech.

This section discusses the additional tasks that must be carried out by language generation in a data-to-speech context. First, Section 1.3.1 offers a brief overview of current methods to produce speech output. In Section 1.3.2 the role of prosody in speech is explained, and it is argued that in order to achieve a good prosodic output quality, linguistic information from the language generation component should be used. Finally, Section 1.3.3 discusses some possibilities for fitting prosody computation into the architecture of a data-to-speech system.

1.3.1 Speech generation methods

The most straightforward way to provide a system with speech output is to simply record all utterances that one wants the system to be able to pronounce, and then play them back as required. Of course, this approach is impracticable in all but the simplest of applications, as it will work only for a limited number of sentences, which are exactly known beforehand. A more realistic solution is to use *phrase concatenation*, where pre-recorded phrases existing of one or more words are played back in different orders to form complete utterances. The key merit

of the technique is that it becomes possible to generate sentences that have never been produced as such by any human speaker, but with a quality approaching natural speech. Phrase concatenation is used in several commercial applications such as travel information services (see e.g., Aust et al. 1995), telephone banking systems, and market research tele-services. A major drawback of this technique is that it is practical only if the application domain is limited and remains rather stable.

A method which has none of the disadvantages inherent to the record-and-play-back scheme is that of full-fledged *speech synthesis*. The most common form of speech synthesis is concatenative synthesis. In this approach, a database of very small speech segments is used (usually phonemes or diphones, see Section 2.6.1). Utterances are formed by concatenating the required segments and applying different forms of signal processing to the result. This method offers unlimited flexibility: it is not necessary to restrict the number of possible sentences that can be generated by the application, and addition of new material to be pronounced presents no problem. Unfortunately, there is a price to be paid for this flexibility. Although speech technology has reached the stage where synthesised speech has a high degree of intelligibility, in general the speech still sounds quite unnatural.⁴

1.3.2 The importance of prosody

The naturalness and intelligibility of speech output is influenced by different factors, one of which is *prosody*.⁵ The term ‘prosody’ is generally used to refer to variations in pitch, loudness, tempo and rhythm within an utterance. These variations are determined mainly by *accentuation* and *phrasing*. Some of the words in an utterance are emphasised by pronouncing them with a pitch change. These words are said to be accented. In addition, most utterances are divided into (intonational) phrases, the boundaries of which are often marked by a pause, a rise in pitch and the lengthening of pre-boundary speech sounds.

In natural speech, the prosody of an utterance varies depending on several factors such as the syntactic structure of the utterance and its linguistic context. In Dutch and English, usually the rightmost word in a syntactic phrase is accented, but this default may be changed by contextual factors: if a word expresses information that is already part

⁴For an on-line comparison of speech synthesis systems in various languages, see <http://www ldc.upenn.edu/lts/>.

⁵The user’s comprehension of a spoken text is also influenced by text features such as sentence length and complexity. This issue is not discussed in this thesis; but see Petrič and van den Bergh 1993 for an overview of text features that affect listening performance. The usability of different types of spoken route presentations in the OVIS system (see Chapter 5) has been tested by Claassen 1998;1999;2000.

of the linguistic context, it is not accented. This can be illustrated by example (4), taken from Sproat 1995, where the word *dogs* will often be deaccented because of the earlier reference to the concept ‘dog’. (In this and later examples, accented words are printed in small capital letters.)

(4) My SON wants a DOG, but I am ALLERGIC to dogs.

On the other hand, words expressing information that is *contrastive* are always accented, even if the information they express has been mentioned previously. A well-known example is given in (5). Here, both pronouns receive a contrastive accent, even though they refer to entities that have already been mentioned. The issue of contrastive accentuation of ‘given’ items is discussed in depth in Chapter 3.

(5) John insulted Mary and then SHE insulted HIM.
(Lakoff 1971:333)

Finally, the placement of phrase boundaries within an utterance mainly depends on syntax. An example is shown below in (6), adapted from Sanderman 1996. To indicate that the phrase *with the stick* is used as an adverbial with the verb *hit*, a phrase boundary (indicated by a slash) may be placed after the word *dog*, as in (6)a. On the other hand, if *with the stick* modifies the noun *dog*, as in (6)b, such a phrase boundary is less appropriate.

(6) a. The man hit [the DOG] / with the STICK
b. The man hit [the DOG with the STICK]

In order for speech output to sound natural, the prosodic variations caused by syntactic and contextual factors must be taken into account. This presents a major problem for the conventional approach to phrase concatenation, since in this approach all the necessary words and phrases are recorded once and thus have a fixed prosody, making contextual variation impossible. In contrast to the conventional approach, the IPO approach to phrase concatenation (discussed in detail in Section 2.6.2; see also Klabbbers 2000) does take prosodic variation into account. This is done by recording different prosodic versions for otherwise identical words and phrases. For choosing the correct version during speech generation, this technique depends on the presence of *prosodic markers* in the text to be pronounced. These markers indicate which words in the text should be accented, and where phrase boundaries should occur. For prosody generation in speech synthesis, we see the same dependency: abstract prosodic markers in the text to be pronounced are required as input for the intonational and durational models of the speech synthesiser.

This dependency of speech generation on abstract prosodic mark-

ers, indicating the location of accents and phrase boundaries, raises the question where these markers should come from. This issue is discussed in Section 1.3.3.

1.3.3 Prosody computation: an additional task

Since most language generation systems produce plain written text, the most obvious way of pronouncing this text is by using a *text-to-speech* system. Such systems take plain text as input, then perform some form of text analysis in order to determine the abstract prosodic properties of the text, and finally put the result of this analysis through a speech synthesiser for pronunciation. The main drawback of this scheme is that valuable linguistic information that is present in the language generation system remains unused, even though the prosodically relevant information that can be obtained through text analysis is generally much less reliable and detailed. By exploiting the information from the language generation component, in principle a higher prosodic quality can be obtained than by using a plain text-to-speech system.

In order to exploit this information, different solutions are possible. One solution is to have a monolithic architecture, where language and speech generation are closely integrated. This design may be efficient, because all relevant information is directly available, but it has the disadvantage that language and speech generation are so closely intertwined that it is impossible to reuse either component in another system. An example of such an intertwined architecture is the SSC (Speech Synthesis from Concept) system proposed by Young and Fallside 1979. In this system, the computation of prosody is done during speech generation. The two tasks are inseparable, and fully dependent on the specific input (a full syntactic structure for an utterance) provided by the preceding language generation component.

An alternative solution, proposed by McKeown and Pan 1997, is to have an architecture in which language and speech generation are independent modules which are interfaced by a general prosodic component. The advantage of such an architecture is that the language and speech generation modules are reusable for different applications, and that the intermediate prosody component can (in principle) be used to couple different language and speech generation components. However, in practice the usability of such a prosodic component may be restricted by the variety of linguistic information provided by, and representation formalisms used in, current language generation systems. Given this variety, a separate pre-processing module will be required for almost every language generation system the prosodic component is to be coupled with.

The D2S approach is somewhere in between. In D2S, language and

speech generation from separate, reusable modules, but prosody computation is performed within the language generation module. In line with this architecture (discussed in Section 2.2), in this thesis prosody computation is regarded as a step in the language generation process, performing the final task of adding prosodic markers to the generated text.

1.4 Research aims

This thesis deals with language generation in the context of data-to-speech, and in particular in the context of the D2S system. As argued in the previous section, prosody computation is an important task in data-to-speech. One of the distinguishing characteristics of D2S is that prosody computation is carried out in a fairly sophisticated manner, using information from the language generation module. For instance, based on a model of the linguistic context, certain words are deaccented because they express information that is assumed to be already known to the hearer. However, one aspect of accent assignment remained somewhat underdeveloped in the original language generation module of D2S: the assignment of contrastive accents. Many researchers have noted that to achieve natural prosody, it is necessary to assign an accent to words expressing contrastive information. However, the notion of contrast has long defied formalisation. As a consequence, most speech generation systems do not incorporate a method for the automatic detection of contrast. This thesis examines how information about the linguistic context of an utterance can be used to determine which of its words and phrases express contrastive information.

Another task for which linguistic context is highly relevant, and which is central to all forms of language generation, is the generation of referring expressions. As will be shown in Chapter 2, the context model of D2S is used for determining the type of referring expressions; the task of determining the content of these expressions, however, received less attention in the original language generation module. In other words, D2S originally did not have a principled way of selecting the properties to be included in a definite description. In the literature several algorithms dealing with this issue have been put forward, but none of these explicitly take the linguistic context into account. In this thesis, it is investigated how definite descriptions can be generated in a context-sensitive and efficient fashion.

Finally, the issues mentioned above are examined for language generation in both a monologue and a dialogue context. Thus, the aims of the research presented here can be summarised as follows:

- To determine how information on the linguistic context can be used to establish the placement of contrastive accents;
- To determine how information on the linguistic context can be used for the generation of definite descriptions;
- To investigate the effect of a dialogue context on the above-mentioned issues.

The research results to be presented are based on theoretical insights and new experimental data. They do not depend on any particular generation system, but D2S will serve as a framework for implementation of these results in various applications. Testing the usefulness of D2S, originally developed as part of one specific system, for a wider class of applications, can be seen as an – implicit – additional research aim.

1.5 Overview

In the remaining chapters of this thesis, the following issues are discussed. First, in Chapter 2, a detailed description is given of the D2S system, which forms the *system context* for the research described in this thesis. The architecture of D2S is presented, and each of its modules are discussed in turn: the language generation module, the prosodic component, and the speech generation module. The chapter focuses on the description of the language generation module, presenting among other things a detailed representation and discussion of the main language generation algorithm, which has not been described in detail before. In addition, a detailed account is given of the way in which information about linguistic context and syntactic structure is used for the computation of accents and phrase boundaries in D2S. The D2S system is illustrated using a particular application called *GoalGetter*, a D2S-based data-to-speech system which generates Dutch spoken football reports on the basis of tabular data. The *GoalGetter* system has been developed as part of the research described in this thesis.

Chapter 3 discusses how information about the linguistic context of an utterance can be used to determine which of the words and phrases in this utterance express *contrastive* information. As was briefly pointed out in Section 1.3.2, the accentuation pattern of an utterance depends, among other things, on its linguistic context. Generally, words expressing information that is new to the hearer are accented, whereas words expressing information that is known (for instance, because it was already mentioned), are not. An exception to this general rule is formed by contrastive information: words expressing information that is contrastive are generally accented, even if this information was already mentioned earlier in the discourse. This means that to achieve appropriate pros-

ody, contrast of information must be taken into account. Without it, improper deaccentuation of known but contrastive items may occur. However, the automatic detection of contrast is not a simple task, as contrast is a rather intangible notion. In Chapter 3, some early, informal views on contrast are discussed, as well as some more recent, formal approaches to the prediction of contrast for data-to-speech generation. In addition, a small experiment is described that was carried out to test in what linguistic contexts people have a preference for contrastive accentuation. Finally, a practical, but theoretically motivated way of detecting the presence of contrastive information within D2S is presented.

Chapter 4 presents an algorithm for the generation of *referring expressions* for domain entities, which is one of the essential tasks in language generation. The easiest way of referring to an entity is by using its name. However, proper names are not always available, and in those cases a description of the entity is required. The creation of descriptions may also be required for stylistic reasons, e.g., to avoid repeating a proper name over and over again. To obtain a principled method for the context-sensitive generation of descriptions, an existing algorithm for the generation of definite descriptions has been modified so that linguistic context is taken into account. The basic idea underlying the new algorithm is that a definite description refers to the *most salient* element satisfying the descriptive content. A method for determining salience on the basis of linguistic context is proposed, and it is shown that taking salience into account allows for the generation of reduced descriptions, which still uniquely identify the intended referent in the given context. In addition, a description is given of a small experiment that was carried out to test the hypotheses underlying the revised algorithm. Finally, it is shown that by some simple modifications the algorithm can be used for the generation of other types of referring expressions, such as pronouns and relational descriptions (which describe an object in terms of its relation to another object).

Chapter 5 discusses language generation in a *dialogue context*. It describes how, as part of the research presented here, the language generation module of D2S was adapted for use in a spoken dialogue system called OVIS, which provides information on train tickets and time tables. In addition, the chapter presents an informal discussion of the relation between linguistic context and information status in dialogue. In a dialogue, it cannot be assumed that all information presented by one speaker is immediately 'known' to both speakers. First, they have to signal their mutual understanding of the information. The relevance of this observation for accentuation and referring expression generation

is discussed, based on examples from human-human dialogues. From these it appears that, as soon as both speakers agree on what is in the linguistic context, they can both refer to this information in the same way as in a monologue. However, when a speaker is not sure of what the other has said, he or she cannot use a reduced or deaccented description to refer to the information provided by the other. In human-machine dialogues, such communication problems are very common. The chapter finishes with a description of a corpus study, investigating which linguistic and prosodic cues people use to signal the presence or absence of communication problems in human-machine interaction.

Finally, in Chapter 6 the issues discussed in the preceding chapters are recapitulated and several pointers to future research are given.

Description of D2S

2.1 Introduction

This chapter presents a detailed description of the D2S system developed at IPO. D2S is a generic system that can be used for the construction of data-to-speech systems for various domains and languages. The most important characteristic of data-to-speech is that it combines language and speech generation. An obvious way of combining the two is to have two separate modules for language and speech generation whose interface consists of plain text. A serious drawback of such an architecture is that valuable information for speech generation is lost (Pan and McKeown 1997, Zue 1997). For this reason, in D2S linguistic information provided by the language generation module is used for the reliable generation of prosodic markers, thus improving the prosodic quality of the system's speech output.

D2S is a *hybrid* system, in which some parts of the generation process are based on general, linguistic principles, whereas other generation tasks are carried out using less flexible, application-specific methods. One of the interesting features of the language generation module of D2S is that it does not follow the relatively common pipeline architecture for language generation (Mykowiecka 1991, Reiter 1994, Cahill et al. 1999) in which text and sentence planning precede linguistic realisation. In fact, D2S contains hardly any *global* text planning. In order to maximise variation between generated texts, local conditions determine which sentences can be used properly given the current state of the generation process. (This issue is discussed in Section 2.4.3.) D2S can operate with two different speech generation modules: one based on pre-recorded phrases, which offers high speech quality but is very inflexible, and one based on diphone synthesis, which offers high flexibility but has a lower output quality. What the two have in common is that they can make use

of the prosodic marking provided by language generation to determine prosody.

D2S was initially developed with the construction of the Dial Your Disc (DYD) system (van Deemter et al. 1994, Odijk 1995, van Deemter and Odijk 1997). The DYD-system provides a natural language interface to a music database which contains information about recordings of Mozart compositions, as well as the music recordings themselves. In DYD, natural language is used both for *browsing* through the database and for *presentation* of the results. During browsing, the user can specify filters on the database using typed or spoken natural language, and the system provides spoken natural language feedback. After a filter (e.g., *a sonata played by Uchida*, van Deemter 1996:7) has been applied to the database, the system selects a sample from the set of music recordings that pass the filter. A fragment of this sample recording is played, and the system presents information about the recording in the form of a spoken natural language monologue. The D2S system is a generalised version of the DYD-module that is responsible for the generation of these spoken monologues. By stripping away all application-specific parts from this module, a general framework has been formed that can be used as the basis for a whole range of applications.

In this thesis, two D2S-based applications are discussed. The first application is GoalGetter, a data-to-speech system which generates spoken reports of football matches in Dutch, based on tabular data (Klabbers et al. 1996, Klabbers et al. 1998). GoalGetter is described in detail in the current chapter. It has been developed to demonstrate the wide applicability of the techniques used in D2S, as GoalGetter differs from DYD with respect to both language (Dutch versus English) and domain (football versus Mozart). In addition, D2S has been used for output generation in a Dutch spoken dialogue system called OVIS (Veldhuijzen van Zanten 1998, van Noord et al. 1999, Klabbers 2000). OVIS is a public transport information system which can be accessed via the telephone. The use of D2S (and in particular, its language generation module) in OVIS is discussed in Chapter 5.

In the current chapter, which is largely based on Theune et al. 2000, the GoalGetter system is used to illustrate D2S and the techniques it is based on. Specific aspects of D2S that have been described in earlier publications are topic management (Odijk 1995), prosody computation (Theune et al. 1997), language generation (van Deemter and Odijk 1997) and speech generation (Klabbers 1997;2000). The current chapter, however, gives a detailed overview of D2S as a whole. It focuses on the description of the system's original language generation module, which was the starting point of the research presented in this thesis. It

presents a detailed representation and discussion of the main language generation algorithm, which has not been described in detail before.

The chapter is organised as follows. Section 2.2 gives an overview of the architecture of D2S, together with a brief discussion of its advantages and disadvantages. Section 2.3 introduces the GoalGetter system and gives an example of its input and output. In the subsequent sections, examples from GoalGetter are used to illustrate the workings of the different modules of D2S. First, Section 2.4 describes the language generation module (LGM) of D2S. Then, Section 2.5 explains how prosodic markers are assigned by the prosodic component of the LGM. Finally, Section 2.6 discusses the use of these markers by the speech generation module (SGM) of D2S. The chapter ends with a discussion in Section 2.7.

2.2 The architecture of D2S

The general architecture of D2S is sketched in Figure 2.1. The language generation module of D2S (abbreviated as *LGM*) takes data as input and produces *enriched text*, i.e., text which has been annotated with prosodic markers by the prosodic component of the LGM. This component determines the placement of accents and phrase boundaries in the output text, using the linguistic information that is present in the LGM. The enriched text is then used as input for the speech generation module (abbreviated as *SGM*), which turns it into a speech signal. By using the prosodic information in the enriched text, the SGM is able to generate speech output that has a higher prosodic quality than could be obtained when using a plain text-to-speech system. In text-to-speech systems, the linguistic information that is relevant for prosody must be obtained through linguistic analysis of the input text. Such an analysis may yield unreliable and incomplete results, which has a negative impact on the prosodic quality of the speech output. In contrast, D2S makes use of the combination of language and speech generation by exploiting the linguistic information which is present in the language generation component for prosody computation.

In D2S, language and speech generation form separate, reusable mod-

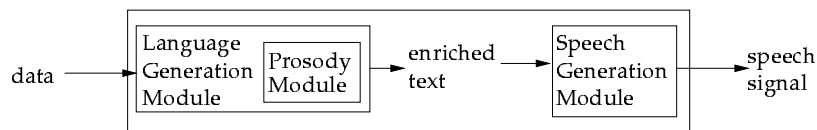


FIGURE 2.1: Global architecture of D2S.

ules, which are connected through a prosodic component. The advantage of this architecture is that both language and speech generation are reusable: the language generation module can be used stand-alone (without speech output) or in combination with different speech output methods, discussed in Section 2.6. In their turn, the speech output methods currently employed in D2S can be used in other systems, given that the required prosodic mark-up is provided in the input. Only the prosody module cannot be ported to another system, since it is embedded in the language generation module, with which it shares a mutual knowledge source containing information about the context.

2.3 The GoalGetter system

In this chapter, examples from GoalGetter are used to illustrate D2S and the techniques it is based on. As noted in the introduction, various applications have been developed on the basis of D2S. Of these, GoalGetter is the least complex due to its limited domain. This makes it suitable for use as an example, but it will also leave some aspects of D2S (in particular the method of *topic management* employed in the LGM, see Section 2.4.3) somewhat underexposed. This is further discussed in the relevant sections below.

The GoalGetter system generates Dutch spoken summaries of football matches. The data which form the input for GoalGetter are automatically retrieved from *Teletext*, a system with which textual information is broadcast along with the television signal. The information is distributed over various ‘pages’, each filling a screen, which are continuously refreshed and are also available via the Internet. Some pages contain textual information, e.g., news messages, and some contain tables, e.g., weather reports and sports results.

Figure 2.2 shows an example Teletext page with information about two football matches. For each match, information about the home team is shown on the left; information about the visiting team is shown on the right. Behind each team name, this team’s result is shown. Below it, a list is given of all players that scored a goal for this team. The minute in which a certain player scored is given between brackets behind the player’s name. ‘Special’ goals are indicated using a specific marker; for instance, */pen* indicates a penalty. Then, the name of the referee (*arbiter*) and the number of spectators (*toeschouwers*) are given. Finally, for each team a list is given of all players who received a card.¹

¹A player who commits a minor offence receives a yellow card. For a major offence, he receives a red card and is sent off the field. Two yellow cards amount to one red card.

SPORT		VOETBAL	
PTT-TELECOMPETITIE			
NOS-11 668 zo 8 okt 16.11:05			
FORTUNA SITTARD	2	GO AHEAD EAGLES	2
Hamming (17,48)		Schenning (18)	
		Decheiver (65)	
Arbiter:		Toeschouwers:	
Uilenberg		4.500	
Geel:		Marbus	

RODA JC	1	PSV	1
Roelofsen (68)		Cocu (78)	
Arbiter:		Toeschouwers:	
Jol		11.500	
Geel:		Van der Weerden,	
		Faber, Jonk, Numan	
uitslagen 661 / stand 662			

FIGURE 2.2: Teletext page containing data from two football matches. (Arbiter = referee; toeschouwers = spectators; geel = yellow card)

An example output text, describing the first match of Figure 2.2, is given in Figure 2.3 together with its translation. The output text is given in enriched text format, i.e., including prosodic mark-up. Accented words are printed in small capital letters, and phrase boundaries of different strengths are indicated by a number of slashes (/ , // or ///). The marker <new-par> indicates the start of a new paragraph.

Two systems which have the same application domain as GoalGetter, are SOCCER (Andre et al. 1988) and MIKE (Tanaka et al. 1998). However, both SOCCER and MIKE generate *commentaries*, spoken descriptions of image sequences of football scenes, whereas GoalGetter generates *summaries* of football matches, taking tabular information about the match as input. In that respect, GoalGetter is more like the STREAK system (Robin 1994, McKeown et al. 1995), which also generates sports summaries, though the sports domain is basketball instead of football. An important difference between GoalGetter and STREAK is that the latter produces only written, not spoken, output.

An interactive, on-line demonstration of the GoalGetter system can be found at <http://iris19.ip0.tue.nl:9000/>.

Output:

Go Ahead EAGLES / ging op BEZOEK bij Fortuna SITTARD // en speelde GELIJK ///
 Het duel eindigde in TWEE // - TWEE ///
 VIJFENVEERTIG honderd TOESCHOUWERS / kwamen naar "de BAANDERT" ///

<new-par>

De PLOEG uit SITTARD / nam na ZEVENTIEN MINUTEN de LEIDING / door een TREFFER
 van HAMMING ///
 EEN minuut LATER / bracht SCHENNING van Go Ahead EAGLES / de teams op GELIJKE
 HOOGTE ///
 Na ACHTENVEERTIG minuten / liet de AANVALLER HAMMING / zijn TWEEDE doelpunt
 aantekenen ///
 In de VIJFENZESTIGSTE minuut / bepaalde de Go Ahead EAGLES speler DECHEIVER
 de EINDSTAND / op TWEE // - TWEE ///

<new-par>

De wedstrijd werd GEFLOTEN door SCHEIDSRECHTER UILENBERG ///
 Hij deelde GEEN RODE KAARTEN uit ///
 MARBUS van Go Ahead EAGLES / liep tegen een GELE kaart aan ///

Translation:

Go Ahead EAGLES / visited Fortuna SITTARD // and DREW ///
 The duel ended in TWO // - ALL ///
 FOUR thousand FIVE hundred SPECTATORS / came to "de BAANDERT" ///

<new-par>

The TEAM from SITTARD / took the LEAD after SEVENTEEN MINUTES / through a GOAL
 by HAMMING ///
 ONE minute LATER / SCHENNING from Go Ahead EAGLES / EQUALISED the score ///
 After FORTY-EIGHT minutes / the FORWARD HAMMING / had his SECOND goal noted ///
 In the SIXTY-FIFTH minute / the Go Ahead EAGLES player DECHEIVER brought the
 FINAL SCORE / to TWO // - ALL ///

<new-par>

The match was OFFICIATED by REFEREE UILENBERG ///
 He did NOT issue any RED CARDS ///
 MARBUS of Go Ahead EAGLES / picked up a YELLOW card ///

FIGURE 2.3: Example output of the LGM, describing the first match from Figure 2.2. Accents are indicated by small capital letters, phrase boundaries by /, // or ///, and the start of a new paragraph by <new-par>.

2.4 Language generation in D2S

The language generation module of D2S (from now on abbreviated as LGM) was designed for spoken information presentation in situations where the user is likely to hear several presentations in succession. Variation in the generated presentations is important here, and this is reflected in the architecture of the LGM, sketched in Section 2.4.1. In particular, the LGM has no global text planner but instead makes use of a set of syntactic templates with conditions on their use (discussed in Section 2.4.2). Combined with the use of topic information to achieve coherence, these conditions act as a kind of local, reactive planner, as discussed in Section 2.4.3. The selection of syntactic templates and the filling of their slots is discussed in Section 2.4.4; finally, a detailed example is given in Section 2.4.5.

2.4.1 Architecture

The general architecture of the LGM is depicted in Figure 2.4. The module *Generation* contains the basic generation algorithm of the LGM (see Figure 2.9). It takes data from outside the system as input; in GoalGetter these are data concerning the characteristics of a particular football match. Because the Teletext pages provide football results in a fixed format, a simple parser can be used to convert the information they contain into typed data structures, which are used as a basis for generation. Figure 2.5 shows the LGM's input data structure for the first match of Figure 2.2.

In addition to the input data, the LGM also uses *domain data*, i.e., a collection of relatively fixed background data on the relevant domain. In GoalGetter these are data about the football teams and their players,

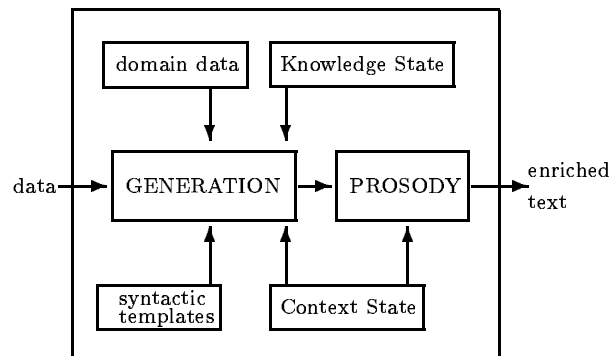


FIGURE 2.4: The architecture of the language generation module.

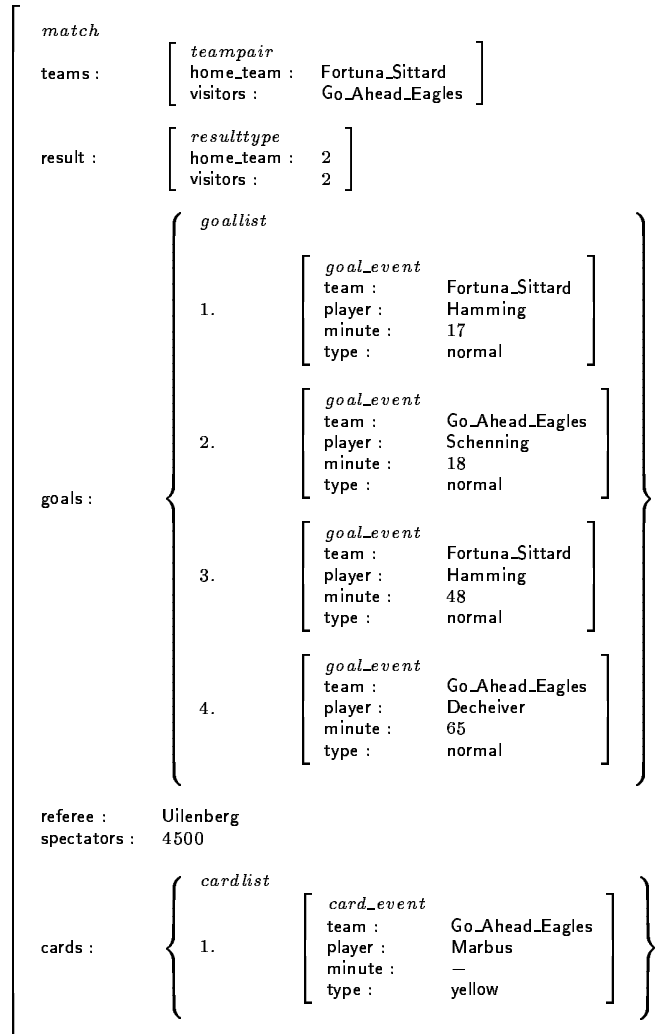


FIGURE 2.5: Data structure representing the first match of Figure 2.2.

such as the home town of each team and the position and nationality of each player. These data are stored in typed data structures for the teams and their players. An example is the feature structure containing information about the team Fortuna Sittard, shown in Figure 2.6. The domain data serve as a supplement to the system's input data from

Teletext, and are used to achieve more variation in the generated texts by providing additional information about the players and teams that are mentioned. For instance, knowledge about the positions of the players provides the system with more possibilities for the generation of referring expressions than if only the Teletext data were available. Examples of the use of domain data in Figure 2.3 are the reference to the stadium of Fortuna Sittard as “*de Baandert*”, the reference to Fortuna Sittard as *the team from Sittard* and the second reference to Hamming as *the forward Hamming*. On the basis of only the input data shown in 2.5 (and derived from the first half of Figure 2.2), “de Baandert” could not have been referred to, and teams and players could only be referred to using their proper names.

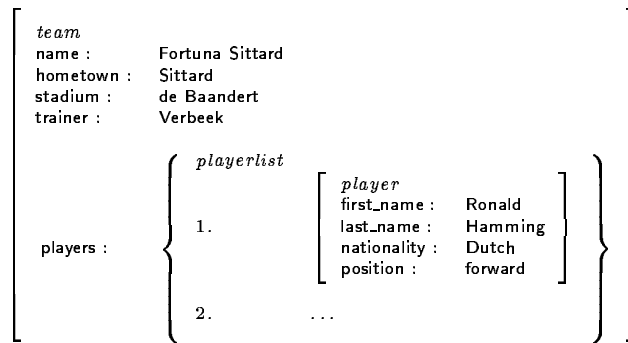


FIGURE 2.6: Background data structure for the team Fortuna Sittard.

The module *Generation* additionally uses a collection of *syntactic templates* to express (parts of) the input data. Syntactic templates contain syntactic tree structures with open slots for variable information. The syntactic information from the templates is used for prosody computation (see Section 2.5), and for checking certain grammatical conditions (see Section 2.4.4). Each syntactic template has a (complex) condition on its use, and the interplay between these conditions during generation determines the structure of the generated text. The form and content of the syntactic templates and their use in generation are discussed in detail below.

During generation, two records are kept. One of them is the *Knowledge State*. It records which parts of the input data structure have been expressed by the system (these are assumed to be *known* to the user) and which parts have not (these are assumed to be *unknown*). The Knowledge State takes the form of a labelling on all fields in the input

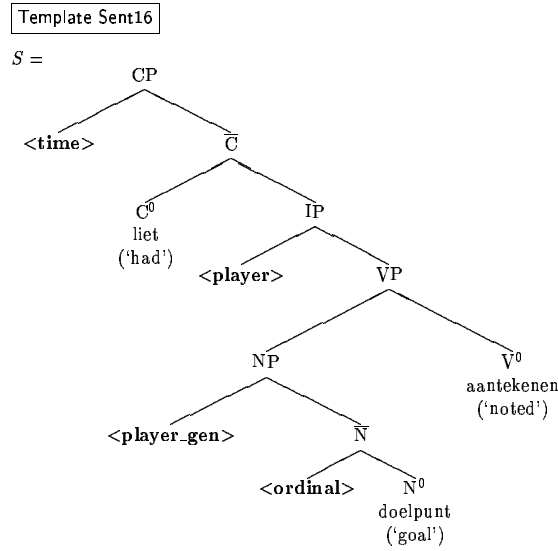
data structure, indicating if their values are known to the user or not. Initially, all fields in the input data structure are labelled ‘unknown’. After generation of a sentence that expresses one or more fields of the input data structure, these fields are labelled as ‘known’. The Knowledge State information is used to guide the selection of templates by the *Generation* module.

In addition to the Knowledge State, there is another record which is kept during generation: the *Context State*, which records various aspects of the linguistic context (i.e., the text that has so far been generated). A central part of the Context State is the *Discourse Model*, which keeps track of the discourse objects that have been mentioned. The information in the Context State is used, among others, during the generation of referring expressions and the computation of prosody. For a detailed discussion of the modelling and use of contextual information in the LGM, see van Deemter and Odijk 1997. Finally, the *Prosody* component computes the prosodic features of each generated sentence.

Due to its use of syntactic templates with associated conditions, discussed in Section 2.4.2, the LGM does not have a pipeline architecture as sketched in Section 1.2.1. In the LGM, three of the main generation tasks distinguished by Reiter and Dale 1997 (outlined in Section 1.2.1) are performed directly through application of the syntactic templates. These tasks are sentence aggregation, lexicalisation, and linguistic realisation. No specific algorithms or system components are required for these tasks. The generation of referring expressions, however, does constitute a separate task in the LGM. It is used to fill the open slots in the templates, and is thus a part of the template application procedure (see Section 2.4.5 below). Unlike most NLG-systems, the LGM does not have global discourse planning. Instead, it uses a form of local planning, discussed in Section 2.4.3 below. Finally, in principle the LGM allows for a simple form of content determination. This is used in the DYD-system, where the user can indicate whether he or she wishes to hear a short or a long text presentation. For the generation of a short text, the system uses only a subset of all available templates. Content determination in the LGM is not discussed in this chapter, since in the GoalGetter system no content determination is performed: all input data are expressed in the output text.

2.4.2 Syntactic templates

One of the main characteristics of the LGM is the usage of *syntactic templates*. Each syntactic template can be used to express one or more parts of the system’s input data structure. Figure 2.7 contains an example template from GoalGetter, which has been used to generate the



$E = \text{time} \leftarrow \text{ExpressTime}(\text{currentgoal.minute})$
 $\text{player} \leftarrow \text{ExpressObject}(\text{currentgoal.player}, \text{nom})$
 $\text{player_gen} \leftarrow \text{ExpressObject}(\text{currentgoal.player}, \text{gen})$
 $\text{ordinal} \leftarrow \text{ExpressOrdinal}(\text{ordinalnumber})$
 $C = \text{Known}(\text{match.teams}) \wedge$
 $\text{currentgoal} = \text{First}(\text{unknown}, \text{match.goals}) \wedge$
 $\text{GoalsScored}(\text{currentgoal.player}) > 1 \wedge$
 $\text{currentgoal.type} \neq \text{owngoal}$
 $T = \text{'game_course'}$

FIGURE 2.7: Syntactic template used for the generation of the sixth sentence of Figure 2.3, *Na achtenveertig minuten liet de aanvaller Hamming zijn tweede doelpunt aantekenen* (“After forty-eight minutes the forward Hamming had his second goal noted”). (CP = Complementiser Phrase, IP = Inflectional Phrase)

sixth sentence of the example text in Figure 2.3. Formally, a syntactic template σ is a quadruple $\langle S, E, C, T \rangle$, where S is a syntactic tree (typically for a sentence) with open slots in it, E is a set of links to additional syntactic structures which may be substituted in the gaps of S , C is a (possibly complex) condition on the applicability of σ and a T is a set of topics. Let us discuss the four components of the syntactic templates in some more detail, beginning with the *syntactic tree* S .

All interior nodes of the syntactic tree structures S are labelled by

non-terminal symbols, while the nodes on the frontier are labelled either by terminal or non-terminal symbols. The non-terminal nodes on the frontier are the gaps which are open for substitution. The syntactic trees in the templates bear a certain resemblance to the initial trees of Tree Adjoining Grammar (TAG, Joshi 1987), but a notable difference with TAG trees is that the latter are generally ‘minimal’, i.e., only the head of the construction is lexicalised and the gaps coincide with the arguments of the head, whereas the syntactic trees in the templates may contain more words, often in order to express collocations (groups of words with a frozen meaning). Examples of collocations occurring in the GoalGetter templates are *een doelpunt laten aantekenen* (“have a goal noted”) (as in Template Sent16) and *de leiding nemen* (“take the lead”).

The second element of a syntactic template is E: *the slot fillers*. Each open slot in the tree S is associated with a call of a so-called Express function, which generates the set of possible slot fillers for the given gap.

The third ingredient is C: *the condition*. A template σ is applicable if and only if its associated condition is true. Two kinds of conditions can be distinguished: (i) conditions on the Knowledge State and (ii) linguistic conditions. Conditions of the former type state things like “ X should not be conveyed to the user before Y is conveyed”. The first two (sub)conditions of Template Sent16 are of this kind. They state that the template can only be used if the competing teams have been introduced to the user (i.e., are known) and the current goal is the first one which has not been conveyed (is unknown). The first condition has to do with the desired global discourse structure. GoalGetter employs the strategy of first presenting general information, and then giving further details. So, the competing teams should be known to the user before the system can describe who scored when. The first condition therefore checks if the teams field of the input match has been labelled ‘known’. The second condition ensures that the template only expresses goals which have not been previously described: the function *First* takes the first *goal-event* from the goals list that is labelled ‘unknown’. If there is no such goal (i.e., all goals have been described), the template is not applicable.

‘Linguistic’ conditions are related to the semantics/pragmatics of the sentence that can be generated from the template, and pose restrictions on the kind of input data to which the template can be applied. The two final conditions on Template Sent16 are of this type. The first of the two says that Sent16 is only applicable if the player of the current goal has scored more than once during the match. Because a sentence of the form *X had his first goal noted* creates the impression that player X has scored at least more than one goal, such a sentence should not be used if X has actually scored only once. The final condition on Template

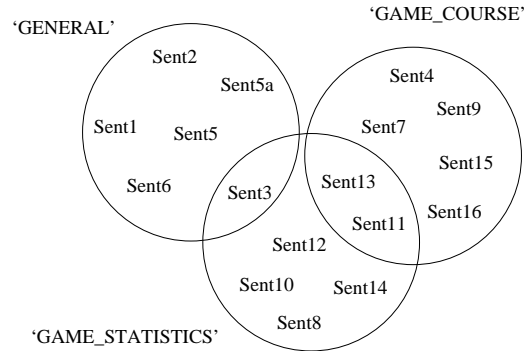


FIGURE 2.8: Topics and templates.

Sent16 states that this template cannot be used to describe an own goal. This restriction is added because using the phrase *having a goal noted* to describe an own goal would give rise to a false conversational implicature (Grice 1975) by creating the impression that the current goal is a normal goal when, in fact, it is not.

Finally, each template σ contains a set of one or more *topics* T . These are labels which globally describe what the syntactic template is about. The LGM algorithm uses the topic information to group sentences together into coherent chunks of text. Each topic has several templates associated with it, and each template is associated with one or more topics. This situation can be illustrated using the simple Venn diagram in Figure 2.8, which represents the topics and templates of the Goal-Getter system. In GoalGetter, which is a relatively small system, there are only three topics: (i) 'general' (giving general information about, for instance, the names of the opposing teams and the final result of the match), (ii) 'game_course' (giving information about events which occurred at a specific time during the match) and (iii) 'game_statistics' (giving details of the match that are not necessarily associated with a specific time, e.g., bookings of specific players). The GoalGetter system currently contains approximately 30 syntactic templates, not all of which are shown in Figure 2.8. Some of these templates belong to more than one topic; for instance, information about red cards (which cause a player to be sent off the field during the game) can either be expressed as part of the 'game_course' or as part of the 'game_statistics'. Similarly, information about the number of spectators of a match may be seen as part of either the 'general' information or the 'game_statistics'.

Not all of GoalGetter's templates are used in each text; for instance,

if no own goals occurred in the match to be described, the templates that express the scoring of an own goal are not used. In addition, each piece of information from the input data structure can typically be expressed using more than one template. For instance, in GoalGetter there are four different templates available to convey information about the referee of a match. The selection of templates and the form of ‘discourse planning’ used in the LGM are discussed in Section 2.4.3 below.

2.4.3 Topics, conditions and coherence

Given a set of syntactic templates that can be used to create sentences expressing parts of the input data, a method is needed for combining these sentences into a coherent output text. Various approaches are possible here, such as the use of an explicit grammar which states where each sentence can occur. A different approach would be to make use of a form of text planning where, before linguistic realisation, the pieces of information to be conveyed are grouped in such a way that a coherent text results. In the LGM, a different approach has been taken, starting as it were from the other side: instead of explicitly specifying in advance where in the output each sentence should occur, it is assumed that in principle all sentences can occur anywhere, but that conditions prohibit their use in some cases. This approach aims at achieving a high degree of variation in the output texts. Variation is important, since the users of typical D2S applications are expected to listen to several texts in succession. If these texts do not show sufficient variation, this will presumably be slightly boring (Odijk 1995).

So, the generated texts must be both varied and coherent. Presumably, the two main requirements for achieving global coherence of a text are (i) the information must be presented in a natural order, and (ii) the information must be presented in natural groupings. To ensure a natural grouping of the sentences in the output of the LGM, the topics associated with the templates are used. Each paragraph in the generated text contains only sentences that have been generated from templates sharing the same topic. In the GoalGetter example in Figure 2.3, the first paragraph corresponds to the ‘general’ topic, the second paragraph is about the ‘game_course’ and the third about ‘game_statistics’.

The ordering of paragraphs in a text and sentences in a paragraph is determined by the conditions on the templates. A template can be used if it belongs to the topic of the current paragraph, and if its conditions evaluate to true given the current Knowledge State. If more than one template is applicable in the current state of the generation process (and this will often be the case), one is chosen arbitrarily. After a sentence has been generated from the chosen template, the Knowledge State is

updated and new templates become applicable. If there are no more applicable templates within the current topic, a new topic must be chosen. There is no *a priori* ordering on the topics; whether a new paragraph can be started given a topic T depends on the applicability of the templates within that topic. If there are no templates associated with T whose conditions evaluate to true in the current Knowledge State, T must be skipped until the Knowledge State has been sufficiently changed for some of its templates to be applicable. Below, the generation algorithm is discussed in detail and illustrated with some examples.

One might argue that the conditions on the templates act as a distributive, reactive planner, in the sense that the conditions are spread across the templates and respond to the current stage of the generation process. This ‘local condition’ approach makes it possible to formulate certain general principles on the presentation of information (e.g., that global information is presented first) without having to specify exactly at which point in the output text each piece of information should be conveyed. This enables the system to achieve a high degree of variation in the generated texts, which is assumed to be pleasant for the hearer in typical D2S applications where several texts are generated in succession.²

The GoalGetter system is not a typical D2S application in the sense that it does not fully exploit the possibilities for variation offered by the LGM’s planning mechanism. The limited variation in the output of GoalGetter is a consequence of the relatively small amount of available data, in combination with the structured nature of a football report (most notably the chronological description of the course of the game). However, other D2S systems have a higher number of topics and templates and a less strict ordering among the topics, thus allowing for much more variation. For instance, the DYD system (van Deemter et al. 1994, Odijk 1995, van Deemter and Odijk 1997) has nine topics which each discuss a different aspect of a Mozart composition, and which may occur in virtually any order, thus reflecting the associative process of describing a composition. In general, it is clear that some types of output text (e.g., descriptions) offer more opportunities for variation than others (e.g., football reports). For each application, the desired amount of variation can be achieved by making the conditions on the templates

²The assumption is that lack of variation may cause the hearer to become slightly bored, especially when listening to a succession of texts. On the other hand, the lack of predictability caused by variation may make it more difficult for the hearer to process the presented information. In some applications, such as the OVIS spoken dialogue system discussed in Chapter 5, variation may therefore be considered undesirable. This point is discussed in Section 5.2.5.

either very global or very strict. Some further examples of templates and their conditions are given in Section 2.4.5. For a more detailed discussion of ‘local planning’ in the LGM, see Odijk 1995.

2.4.4 Algorithm

The LGM generation algorithm is shown in Figure 2.9. Its input is formed by the set of topics (*all_topics*), the set of all syntactic templates (*all_templates*), the initial Knowledge State (*InitialKS*) and the initial Context State (*InitialCS*). In the initial Knowledge State it is recorded that all parts of the input data structure are marked as ‘unknown’, and the initial Context State is simply empty (no discourse entities have been introduced yet). The algorithm starts by initialising two variables, *relevant_topics* and *untried_topics*. The first variable, *relevant_topics*, contains the set of topics that have not yet been used as the basis for a paragraph (none of their templates has been applied yet). The second variable, *untried_topics*, contains the set of topics which the algorithm has not yet tried to use in its current generation round. Both variables are initialised to the set of all topics. Finally, the variables *KS* and *CS* are initialised to *InitialKS* and *InitialCS* respectively.

After initialisation the algorithm gets into its first **while** loop, walking through the set of *untried_topics*. As long as this set is not empty, the algorithm performs the following actions. It starts by randomly picking one of the *untried_topics* and makes it the *current_topic*. Using this topic, it will try to start a new paragraph. First, the variable *possible_templates* is instantiated as the set of those templates that are associated with the *current_topic* and whose conditions evaluate to true in the current Knowledge State. These are the templates that are currently applicable. (It may be that at this point, none of the templates associated with the *current_topic* are applicable yet; in that case, *possible_templates* is empty.) Then the variable *topic_successful* is set to false; this variable records if any template associated with the *current_topic* has been applied successfully (i.e., has produced a sentence). If *possible_templates* is not empty, the second **while** loop is entered. In this loop, the algorithm goes through the set of *possible_templates*, picking one of them at random (*chosen_template*) and trying to apply it by giving it as an argument to the function `ApplyTemplate`, which is discussed in detail below. If `ApplyTemplate` is not successful, it outputs nil. In that case, the algorithm removes the *chosen_template* from the set of *possible_templates*, picks another template from this set and tries to apply it. If the sentence produced by `ApplyTemplate` is not nil, the algorithm orders the Speech Generation Module to pronounce it. The Knowledge State is updated so that the data expressed by means of *chosen_template* are marked as

```

Generate(all_topics, all_templates, InitialKS, InitialCS)
relevant_topics ← all_topics
untried_topics ← all_topics
KS ← InitialKS
CS ← InitialCS
while untried_topics ≠ {}
do current_topic ← PickAny (untried_topics)
    possible_templates ← { t ∈ all_templates | current_topic ∈ Topic (t) ∧
        Conditions (t, KS) = true }

    topic_successful ← false
    while possible_templates ≠ {}
    do chosen_template ← PickAny (possible_templates)
        sentence ← ApplyTemplate (chosen_template, CS)
        if sentence = nil
        then possible_templates ← possible_templates − chosen_template
        else Pronounce (sentence)
            KS ← UpdateKS (KS, chosen_template)
            CS ← UpdateCS (CS, chosen_template)
            topic_successful ← true
            possible_templates ← { t ∈ all_templates | current_topic ∈ Topic (t) ∧
                Conditions (t, KS) = true }

        endif
    endif
endif
if topic_successful = false
then untried_topics ← untried_topics − current_topic
else relevant_topics ← relevant_topics − current_topic
    untried_topics ← relevant_topics
    StartNewParagraph
endif
endwhile

```

FIGURE 2.9: The basic generation algorithm of the LGM.

‘known’, and the Context State is updated by (among other things) adding the discourse entities mentioned in the *chosen_template* to the Discourse Model. *Topic_successful* is then set to true, since one of the templates from the *current_topic* has been applied successfully. Now, the set of *possible_templates* is re-instantiated given the updated Knowledge State. (Because the Knowledge State has changed by applying a template, new templates may have become applicable and others unapplicable.) From the new set of *possible_templates*, one template is picked and the procedure starts all over again. This continues until there are no applicable templates left within the *current_topic*. The algorithm then leaves the second **while** loop and checks if the *current_topic* has been successful. If not, it means that nothing has happened; no sentences have been generated. In that case, *current_topic* is removed from the set of *untried_topics*. The set of *relevant_topics* does not change, so that the *current_topic* may be tried again later. The algorithm now picks a new

```

ApplyTemplate(template, CS)
allowed_trees ← {}
sentence ← nil
all_trees ← FillSlots (template, CS)
for each member ti of all_trees do
  if Violate_BT (ti) = false ∧
    Wellformed (UpdateDiscourseModel (CS, ti)) = true
  then allowed_trees ← allowed_trees + ti
  endif
if allowed_trees ≠ {}
then chosen_tree ← PickAny (allowed_trees)
      final_tree ← AddProsody (chosen_tree, CS)
      sentence ← Fringe (final_tree)
endif
return sentence

```

FIGURE 2.10: The function ApplyTemplate.

topic from the remaining *untried_topics*, and tries to start a paragraph with that topic.

If *topic_successful* is true, this indicates that a paragraph of one or more sentences has been generated, so the algorithm now has to start a new paragraph. The *current_topic* is removed from the *relevant_topics*, and the set of *untried_topics* is instantiated as the set of all *relevant_topics*. This means that all topics that were unsuccessful in a previous Knowledge State, can now be tried again as the basis for the new paragraph. The algorithm picks a new topic and continues until there are no *untried_topics* left. This concludes the discussion of the main generation algorithm; we now turn to the function ApplyTemplate, shown in Figure 2.10.

ApplyTemplate attempts to generate a sentence from a template, given the current Context State. This is done by gathering all sentences that result from all relevant slot fillings of the template, and then filtering out those sentences that violate the Binding Theory (Chomsky 1981) or cannot be used to properly update the Discourse Model. From the remaining sentences, one is picked at random. More specifically, ApplyTemplate works as follows. First, it calls the function FillSlots (*template*, *CS*) to obtain the set of all possible trees that can be generated from the template, using all possible combinations of slot fillers generated by the Express functions associated with the slots in the templates. Typically, there are several possible slot fillings for each slot in a template. For instance, a person may be referred to using a proper name, a definite description or a pronoun. As a consequence, FillSlots

typically returns a set of several trees, all expressing the same piece(s) of information albeit in different ways. For each tree in this set it is checked (*i*) whether it obeys the Binding Theory, and (*ii*) if updating the current Discourse Model with the discourse entities from the sentence gives a well-formed result. The system filters out those trees that do not meet these conditions, for instance because they contain a non-pronoun in a bound position (Chomsky 1981; see the example below) or a pronoun that does not have an accessible antecedent in the preceding discourse. One of the remaining trees is then arbitrarily selected and sent to the Prosody module, where its prosodic properties are computed as described in Section 2.5. `ApplyTemplate` then returns the sentence consisting of the tree's terminal nodes plus their prosodic markings (i.e., the 'fringe' of the tree).

Like the random choice of topics and templates, the making of an arbitrary choice from the suitable trees is motivated by the need for variation in the generated texts. Of course, this strategy still leaves room for improvement, for instance because random choices do not guarantee optimal variation. Finally, note that the 'generate and test' strategy employed at several stages of the generation process does not lead to inefficiency, because the LGM is written in a programming language that uses *lazy evaluation*: expressions are only evaluated when necessary.

2.4.5 Example

In this section, the workings of the generation algorithm are illustrated using (parts of) the text in Figure 2.3 as an example. For ease of exposition, throughout this section reference is made only to the English translation of the output generated by `GoalGetter`.

After initialisation of *relevant_topics* and *untried_topics* to the set {'game_course', 'game_statistics', 'general'}, and initialisation of the Knowledge State, the algorithm starts by randomly picking a topic from *untried_topics*. Assume the selected topic is 'game_course'. Now *topic_successful* is set to false (no sentence for this topic has been uttered) and the set *possible_templates* is constructed of all templates which are associated with 'game_course' and whose conditions are true given the current Knowledge State, which says that all parts of the input data structure are still unknown to the user. In this case, the set of *possible_templates* turns out to be empty: there are no 'game_course' templates that are applicable in the initial Knowledge State, when no information about the match has been conveyed yet. This is because all templates in 'game_course' have as their condition that they can only be used if the competing teams are known to the user: before providing details about which player did what during the match, the teams should

have been introduced. Because there are no applicable templates, the ‘game_course’ topic cannot be used for the first paragraph. This means that the attempt with this topic has finished without being successful, so after having removed ‘game_course’ from the *untried_topics* the algorithm starts a new generation round. Although ‘game_course’ remains a relevant topic (nothing has been said about it yet), this time the algorithm can only choose from the two topics which are still untried, ‘general’ and ‘game_statistics’. Assume that now the ‘general’ topic is picked. For explanatory purposes, let us assume that this topic only contains the three templates that have been used to generate the first three sentences of the text in Figure 2.3. (This is a severe simplification.) These templates, Sent1, Sent2 and Sent3, are shown in an abbreviated form in Figure 2.11. Instead of showing the full syntactic trees of the templates, only the ‘flat’ sentences with the slots are shown (in translation); in addition, the topic information and the calls of the Express functions have been left out, and the conditions have been simplified.

For the ‘general’ topic, in the initial Knowledge State the set of *possible_templates* is not empty: it contains both Sent1 and Sent2, which do not require any information to be known before being applied. One of the two templates is chosen at random; this happens to be Sent1. This template can be successfully applied by filling the <team1> slot

Template Sent1

$S = \langle \mathbf{team1} \rangle$ visited $\langle \mathbf{team2} \rangle$ and drew.
 $C = \mathbf{Unknown}(\mathit{match.teams}) \wedge$
 $\mathit{match.result.home_team} = \mathit{match.result.visitors}$

Template Sent2

$S = \langle \mathbf{match} \rangle$ ended in $\langle \mathbf{result} \rangle$.
 $C = \mathbf{Unknown}(\mathit{match.result})$

Template Sent3

$S = \langle \mathbf{spectators} \rangle$ visited $\langle \mathbf{stadium} \rangle$.
 $C = \mathbf{Unknown}(\mathit{match.spectators}) \wedge$
 $\mathbf{Known}(\mathit{match.teams})$

FIGURE 2.11: Abbreviated templates from the ‘general’ topic. The syntactic structures, the topics, and the calls of the Express functions, used to fill the gaps, are omitted.

with the name of the visiting team and the <team2> slot with the name of the home team. (The ApplyTemplate function is illustrated in more detail below.) After the template has been applied, the resulting sentence (the first sentence of Figure 2.3) is pronounced by the SGM, and the Knowledge State is updated with the information that the teams field of the match is now known to the user. The Context State is updated as well, among other things by extending the Discourse Model with discourse entities corresponding to the teams and the match. Then *topic_successful* is instantiated as true (the first sentence of a paragraph has been generated) and the set of *possible_templates* is computed anew, given the updated Knowledge State. Because the competing teams are now known, Sent1 can no longer be used. Sent2 is still applicable because the result of the match has not been explicitly conveyed. In addition, Sent3 has now become applicable because the teams are known. Therefore, *possible_templates* = {Sent2, Sent3}. Now assume that Sent2 is chosen to be applied. Its first slot is filled with an expression for the match. In this case, the definite description *the duel* is used, which is generated using the function ExpressObject, discussed below. The second slot of the template is filled with a tree expressing the result of the match. After application of this template, the result of the match is marked as known. This means that Sent2 is no longer applicable; the only template left is Sent3. After this template has been applied successfully (using the domain information that the stadium of Fortuna Sittard is called “de Baandert”), no more applicable templates are left within the topic, so the paragraph is finished. ‘General’ is removed from the *relevant_topics*, and the *untried_topics* are instantiated as the *relevant_topics*, i.e., {‘game_course’, ‘game_statistics’}. The algorithm now starts a new paragraph, which will be used to illustrate the working of ApplyTemplate and the Express functions.

For the new paragraph, the algorithm now picks ‘game_course’ from the *untried_topics*. Because the competing teams have been introduced in the previous paragraph, this time there are several templates that can be applied given the current Knowledge State. One of them is picked at random and used to generate the fourth sentence of Figure 2.3: *The team from Sittard took the lead after seventeen minutes through a goal by Hamming*. Consequently, the Knowledge State is updated to reflect the fact that information about the first goal has been conveyed. In addition, the Discourse Model is extended with entities corresponding to the phrases *the team from Sittard*, *Hamming* and *after seventeen minutes*. They receive an index indicating which parts of the data structure they refer to. Now the system goes on and attempts to convey the second goal scoring event. It cannot use the same template as the one

used for the fourth sentence, since the second goal scoring event of the current match does not make one team take the lead. Instead, a template is used that is applicable if the scores of two teams are equalised: *One minute later Schenning from Go Ahead Eagles equalised the score.* A noteworthy aspect of this sentence is the time expression that is used. In GoalGetter, there are many possible expressions for the time at which a goal was scored, but one type of expression is particularly interesting, viz. the expression *<N> minute/minutes later*. This expression can only be used if the Discourse Model contains an appropriate reference time, i.e., if the most recent time expression in the Discourse Model is an *explicit* one. The variable *<N>* is then a cardinal expression for an integer which equals the time value of the current event minus the time value of its reference time. Since the current Discourse Model contains an appropriate reference time entry, viz. the expression *after seventeen minutes*, the expression *one minute later* can be used here.

Now a sentence must be generated to describe the third goal of the match. To do this, Template Sent16 (shown in Figure 2.7) is selected. As the reader can verify, all conditions associated with this template are met. After having selected the template, the system attempts to generate a sentence from it using the function *ApplyTemplate* (shown in Figure 2.10), which will now be illustrated in some detail. *ApplyTemplate* first calls *FillSlots* to obtain the set of all possible trees that can be generated from the template, using all possible combinations of slot fillers generated by the associated *Express* functions. Let us start with the first slot, *<time>*. The function *ExpressTime* can generate several time expressions, but one of them is not allowed given the current context: since the most recent time expression in the Discourse Model (*one minute later*) is not explicit, it cannot serve as a reference time for a second expression of the form “*<N> minute/minutes later*”. Such an expression is therefore not allowed here, and *ExpressTime* only returns two possible slot fillings: the explicit time expressions *in the forty-eighth minute* and *after forty-eight minutes* (which are taken to be synonymous here, although strictly spoken this is not true).

The second slot to be filled is the *<player>* slot. The function *ExpressObject*, shown in Figure 2.12, is called to generate an expression (in the nominative case) for the player who scored the third goal (Hamming). Again there are in principle several options: a player can be described using a proper name, a pronoun, a definite description, or an appositive that combines a definite description and a proper name. To generate these different types of expression, *ExpressObject* calls several sub-functions and returns the set of all expressions produced by these functions. In the current example, this set consists of the proper name

```

ExpressObject(r, case)
PN ← MakeProperName (r, case)
PR ← MakePronoun (r, case)
DD ← MakeDefiniteDescription (r, case)
AP ← MakeAppositive (r, case)
trees ← PN ∪ PR ∪ DD ∪ AP
return trees

```

FIGURE 2.12: The function ExpressObject.

Hamming, the pronoun *he*, the definite description *the forward* and the appositive *the forward Hamming*. Note that there are two attributes that could in principle have been used in the definite description of Hamming, i.e., the player's position and nationality attributes (see Figure 2.6). However, the nationality attribute is not used here, because its value is Dutch, which is not deemed very informative in the current domain.³ Therefore, the MakeDefiniteDescription function only returns a definite description expressing Hamming's position attribute.

The third slot, <player_gen>, must be filled with an expression for Hamming in the genitive case. Because MakeDefiniteDescription and MakeAppositive do not return expressions in the genitive case (as these are not well-formed), the function ExpressObject only returns the proper name *Hamming's* and the pronoun *his*.

Finally, the formation of the filler for the <ordinal> slot, expressing the number of goals the current player has scored so far, requires a bit of computation not indicated in Figure 2.7. This computation yields a positive integer (in the current example, 2), which must then be expressed by its corresponding ordinal (*second*).

<time>	{ <i>in the forty-eighth minute, after forty-eight minutes</i> }
<player>	{ <i>Hamming, he, the forward, the forward Hamming</i> }
<player_gen>	{ <i>Hamming's, his</i> }
<ordinal>	{ <i>second</i> }

FIGURE 2.13: Possible slot fillings for Template Sent16.

Combining all different slot fillings, shown in Figure 2.13, the function FillSlots returns a set of $2 \times 4 \times 2 \times 1 = 16$ trees that can be generated

³In the MakeDefiniteDescription function described here, appropriateness of attributes to be included in a definite description is determined in an *ad hoc* fashion by a few application-specific rules. Chapter 4 discusses a theoretically well-founded alternative for this function.

from Template Sent16 in the current context. For each tree in this set, it is first checked whether it obeys the Binding Theory (Chomsky 1981). This test filters out the trees where the proper name *Hamming's* occupies the <playergen> slot, because the proper name (which is a so-called 'R-expression' in the Binding Theory) is not free in this position, thus violating Principle C of the Binding Theory. Next, it is checked for each of the trees if it can be used to properly update the Discourse Model. This check filters out the trees where either the pronoun *he* or the definite description *the forward* occupies the <player> slot. Both these expressions require an accessible antecedent, which is not available in the current discourse: the antecedent (*Hamming* in sentence four) is not accessible due to the intervening reference to the player Schenning.⁴ This leaves the proper name *Hamming* and the appositive *the forward Hamming* as possible slot fillings for the <player> slot, so the set of *allowed_trees* finally contains trees for the following four sentences:

$$\left\{ \begin{array}{l} \textit{After forty-eight minutes Hamming had his second goal noted,} \\ \textit{After forty-eight minutes the forward Hamming had his second goal} \\ \textit{noted,} \\ \textit{In the forty-eighth minute Hamming had his second goal noted,} \\ \textit{In the forty-eighth minute the forward Hamming had his second goal} \\ \textit{noted} \end{array} \right\}$$

From these remaining trees, one is selected arbitrarily (in this case the second one), and sent to the Prosody module, where its prosodic properties are computed. The fringe of the resulting tree (i.e., the sentence enriched with prosodic markers) is returned to the main algorithm, where the sentence is sent to the SGM to be pronounced, and the Knowledge State and the Context State (including the Discourse Model) are updated accordingly. This ends the illustration of the language generation algorithm; Section 2.5 discusses prosody computation in D2S.

2.5 Prosody computation in D2S

This section shows how the Prosody module determines the location of accents and phrase boundaries in a generated sentence on the basis of both syntactic and semantic information, again using the example sentence *Na achtenveertig minuten liet de aanvaller Hamming zijn tweede doelpunt aantekenen* ("After forty-eight minutes the forward Hamming had his second goal noted") as an illustration. The prosodic rules de-

⁴The definite description *the forward* is not uniquely referring: although Hamming's position does in fact distinguish him from Schenning (who happens to be a midfielder) the hearer cannot be assumed to be aware of this.

scribed in this section are independent of domain and language, within the class of Germanic languages (e.g., English, Dutch, and German). The same set of rules has been used for prosody computation in both GoalGetter and the DYD-system, which differ with respect to language (Dutch versus English) and domain (football versus Mozart).

2.5.1 Overview

Since accentuation is relevant for the placement of phrase boundaries, but not vice versa, the Prosody module starts with computing the accentuation pattern of each sentence, using an algorithm that is based on a version of Focus-Accent Theory (Baart 1987) proposed in Dirksen 1992 and Dirksen and Quené 1993. In Focus-Accent Theory, binary branching metrical trees are used to represent the semantic and syntactic prominence of nodes with respect to pitch accent. In D2S, the metrical tree of a sentence is based on the sentence's syntactic tree.⁵ It is constructed by converting the syntactic tree to a tree that is at most binary-branching and marking its nodes with *focus* markers and *weak* or *strong* labels. The focus markers indicate *information status*. Words and phrases that express new or contrastive information are considered to be in focus, whereas words and phrases that express 'given' information are considered to be out of focus. The weak/strong (w/s) labels in the tree represent the structural prominence of the nodes with respect to accentuation. In other words, the focus marking indicates which words and phrases should or should not receive an accent, while the w/s labelling determines where in each constituent an accent may land. The first is a matter of discourse semantics, while the second is a matter of syntax.

2.5.2 Focus and information status

The focus properties of the nodes in the metrical tree are determined as follows. First, the Prosody module adds an initial, preliminary focus marking to the tree. In this initial state, all major constituents (i.e., all maximal projections of the form XP) are, by default, assumed to be in focus and are marked [+F]. The other nodes are initially not specified for

⁵Having such a direct link between "traditional syntactic structure" and intonational structure is somewhat controversial. Steedman 1990;1996 and others have pointed out that prosodic structure does not always adhere to the traditional subject-predicate division of a sentence. An example from Steedman 1996 is the following:

(7) Q: Well, what about MARY? What does SHE admire?
A: MARY admires / MUSICALS

This example is represented using the notation adopted in this thesis; the placement of accents and phrase boundaries is Steedman's. The prosodic component of the LGM would not produce a phrase boundary in the above example (but it would generate the same accentuation pattern).

focus. After the initial, default assignment of focus markers has taken place, the system tries to determine the *information status* of the words or phrases in the tree.

The notion of information status used in D2S is based on Chafe 1976, who observes that the way information is expressed in spoken language depends on the assumptions the speaker makes about the status of this information in the hearer's current mental model. With regard to this mental model, Chafe distinguishes six different information statuses, two of which influence the accentuation of the information to be expressed. The first is givenness: if the hearer is assumed to be familiar with the information, for instance because it was already mentioned in the linguistic context, the speaker tends to deaccent the words expressing it. If the information is not given, it is new, and may be accented (depending on its position in the sentence and other factors, such as informativeness). The second information status is contrast: Chafe 1976 observes that information which is assumed to be contrastive is always accented by the speaker, even if it is given. Although Chafe distinguishes several other forms of information status, in D2S the notion is restricted to cover only those statuses which influence accentuation: givenness (and its counterpart, newness) and contrast.⁶

First of all, the system checks which words and phrases in the current sentence are *contrastive*. In the current example, these are the phrases referring to the time of the goal (*na achtenveertig minuten*) and to the player who scored it (*de aanvaller Hamming*). This information is in contrast with the time and player of the previous goal.⁷ The contrastive phrases are marked [+C] in the metrical tree, to indicate that they are in focus due to contrast. If a constituent expresses contrastive information, its focus marking cannot be changed, even if it might be regarded as given (for a detailed discussion see Chapter 3).

Next, the system determines which words or phrases in the tree express given information, and which words are 'unaccentable' (e.g., certain function words). On the basis of this information, the initial placement of focus markers in the tree may be altered (except for the [+C] markers). The focus value of a node is changed to [-F] in three cases: (i) if the node directly dominates a word or phrase expressing given information, (ii) if it directly dominates an 'unaccentable' word and (iii) if all the nodes it dominates are marked [-F].

⁶The influence of givenness and contrast on accentuation has been generally accepted, and has been the subject of extensive scientific investigation, e.g., Bock and Mazzella 1983, Brown 1983, Terken and Nootboom 1987, Terken and Hirschberg 1994, and Birner 1994 (givenness); Prevost 1995 and van Deemter 1999 (contrast).

⁷These contrasts are detected using the method described in Chapter 3.

Information from the LGM's Context State is used to determine whether a word or phrase expresses given information. The rules for determining givenness are based on the theory proposed by van Deemter Deemter 1994b, who, like Chafe 1976, distinguishes *object-givenness* and *concept-givenness*. A word or phrase is object-given if it refers to a discourse entity (e.g., a player) that has already been mentioned, and it is concept-given if it expresses a concept (e.g., 'scoring a goal') which has already been evoked earlier in the discourse. In D2S, the Discourse Model can be used to check object-givenness, because it records which entities have so far been referred to by the system. In the example sentence, the NPs *de aanvaller Hamming* ("the forward") and *zijn* ("his") are object-given, because their referent, the player Hamming, was referred to two sentences earlier (see Figure 2.3). The focus marking of the second NP (*zijn*) is therefore changed to [-F], but the marking of the first NP does not change because it is marked as contrastive.

Concept-givenness is determined by checking whether the words and phrases in the current sentence are synonymous or identical to words or phrases that were used earlier in the text, or whether the concept they express *subsumes* another concept that was expressed earlier. An example of concept-givenness through subsumption can be found in the seventh sentence of the example text in Figure 2.3: the concept 'player', which is referred to in the appositive NP *de Go Ahead Eagles speler Decheiver* ("the Go Ahead Eagles player Decheiver"), subsumes the concept 'forward' mentioned in the preceding sentence, and is therefore regarded as given. (Since a forward is a player, mentioning the concept 'forward' is assumed to automatically activate the subsuming concept 'player'.) In D2S, subsumption is currently determined using a small hand-crafted subsumption hierarchy, which is application-specific. In addition, application-specific lists of synonyms are used, recording for instance that the words *treffer* and *doelpunt* are synonyms for *goal*. Following Hirschberg 1992, givenness is related to topic structure, assuming that items only remain given within one topic, which corresponds to one paragraph in the output of D2S.

The example sentence contains two cases of concept-givenness. First, the word *minuten* ("minutes") expresses the same concept as *minuut* ("minute"), occurring in the previous sentence, and is therefore defocused. Second, the concept expressed by the collocation *een doelpunt laten aantekenen* ("having a goal noted") subsumes the concept expressed by the collocation *op gelijke hoogte brengen* ("equalise"), a specific instance of scoring a goal expressed in the previous sentence. The words *doelpunt*, *liet* and *aantekenen* are therefore regarded as expressing given information and marked as [-F].

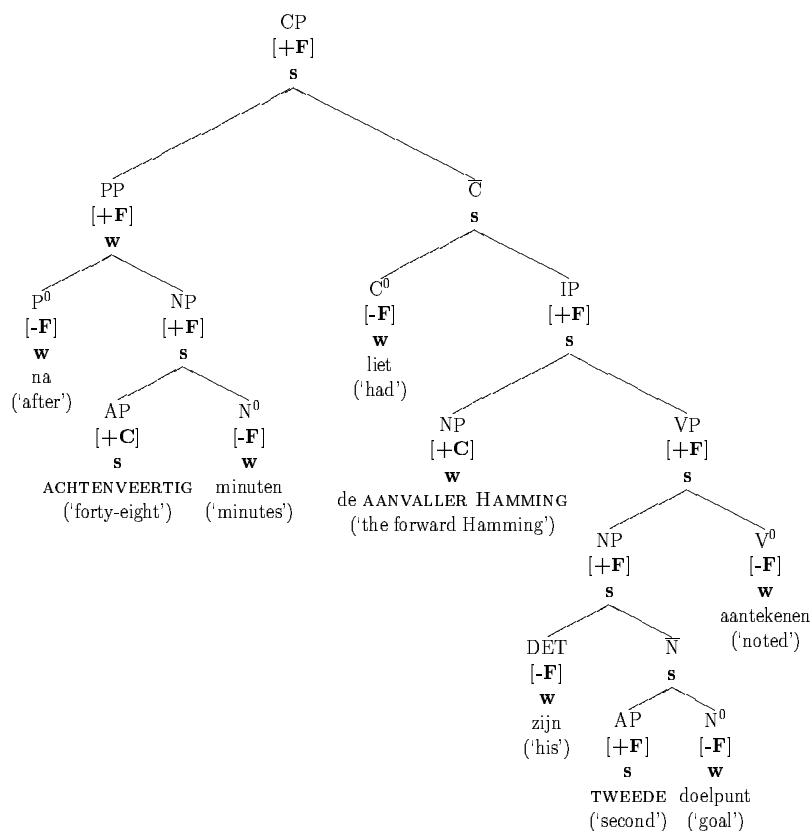


FIGURE 2.14: Final metrical tree of the sixth sentence of Figure 2.3.

2.5.3 Weak and strong nodes

The weak/strong labelling of the metrical tree nodes, which ultimately determines on which words an accent may land, depends both on the structure of the tree and on the focus properties of its nodes. In Dutch, like in English, normally the left node of two sisters is weak and the right node is strong. If the structurally strong node is marked [-F] while the structurally weak node is not, the weak/strong labelling is switched due to the so-called Default Accent rule (Ladd 1980, Baart 1987). In Figure 2.14, showing the complete metrical tree of the example sentence, this has occurred in three cases: (i) for the AP *achtenveertig* and the

defocused N⁰ *minuten*, (ii) for the AP *tweede* and the defocused N⁰ *doelpunt*, and (iii) for the NP *zijn tweede doelpunt* and the defocused V⁰ *aantekenen*.

When the metrical tree is complete, the focus markers indicate which constituents should be accented, and the weak/strong labelling indicates on which words the accent may land. The actual accentuation algorithm can therefore be very simple: each node that is marked [+F] launches an accent, which trickles down the tree along a path of strong nodes until it lands on a terminal node, dominating a word.

In the example, the accents launched by CP, IP and VP all coincide with the accent launched by the NP node of *zijn tweede doelpunt*, finally landing on the word *tweede*, and not on the defocused N⁰ *doelpunt*. (If the N⁰ node had not been defocused, the accent would have landed on *doelpunt*, and the word *tweede* would still have been accented due to the accent launched by the AP node.) Since the nodes dominating *liet* and *aantekenen* are weak, no accent trickles down to them, and because they are marked [-F] they do not launch an accent themselves. The PP node dominating the phrase *Na achtenveertig minuten* does launch an accent, which trickles down to the NP *achtenveertig minuten*, where it coincides with the accent launched by the NP itself. Within the NP, the accent goes to the left because the right node dominating *minuten* has been defocused, so it ends up on the word *achtenveertig*. Finally, the appositive, contrastive NP *de aanvaller Hamming* consists of two NPs (not shown in the tree due to space restrictions), both of which launch an accent that trickles down to their head nouns.

2.5.4 Phrase boundaries

After accentuation, phrase boundaries are assigned. Currently, three phrase boundary strengths are distinguished.⁸ The strongest of the three is the *sentence-final* boundary (///). Next comes the *major* boundary (/ /), which follows words preceding a punctuation symbol other than a comma (e.g., “;”) and sentence-internal clauses (i.e., a CP or IP within a sentence). Finally, a *minor* boundary (/) follows words preceding a comma and constituents meeting the following conditions: (i) the constituent has sufficient length (more than four syllables), (ii) the constituent on its right is an \bar{I} , a \bar{C} or a maximal projection, and (iii) both constituents contain at least one accented word. This is a slightly modified version of a structural rule proposed by Dirksen and Quené 1993. In the present example only the PP *Na achtenveertig minuten* and the

⁸In longer texts, containing more complicated constructions, it might be desirable to distinguish more levels. Sanderman 1996 proposes a boundary depth of five to achieve more natural phrasing.

NP *de aanvaller Hamming* meet this condition and are therefore followed by a minor phrase boundary. Since the example sentence contains no punctuation and consists of just one clause, the only other phrase boundary is the sentence-final one.

2.5.5 Evaluation

The accentuation algorithm of D2S was formally evaluated for Dutch in a small-scale experiment (Nachtegaal 1997). In the experiment, recordings were made of non-professional speakers of Dutch who read aloud the plain text versions of texts generated by the LGM of GoalGetter. The texts contained sentences which were structurally similar to those of the example text given in Figure 2.3. ‘Expert listeners’ were presented with the recordings and indicated on which words they heard an accent. The accentuation patterns produced by the speakers were then compared to those generated by the system. The results of this comparison showed that the number of words on which the accentuation by GoalGetter deviated from the accentuation by the speakers was very small. For details, see Nachtegaal 1997.

For the algorithm determining the placement of phrase boundaries no formal evaluation has taken place yet. However, the general prosodic quality of the output of D2S has been informally compared with the output of the two best text-to-speech systems currently available for Dutch (according to the anonymous evaluation in Sluijter et al. 1998). One of the two systems employs the same speech synthesis as used in GoalGetter (see Section 2.6.1); the other system employs a different kind of speech synthesis. The two text-to-speech systems were used to pronounce some texts generated by the GoalGetter system (plain text version). The prosodic quality of the speech output was then compared to that produced by GoalGetter. This revealed two main flaws displayed by both text-to-speech systems, which made their output sound somewhat less natural than that of GoalGetter. First, the placement of phrase boundaries by the two text-to-speech systems was less adequate than in D2S: several obvious phrase boundaries were missing (e.g., between conjugated clauses), or misplaced (e.g., between an adjective and the NP it modified). Second, both systems failed to perform deaccentuation even in the simplest cases, like the second occurrence of the word *kaart* (“card”) in the following example:

- (8) BLOM *gaf* COCU *een* GELE KAART.
 VOS *kreeg een* RODE *kaart* / ?? VOS *kreeg een* RODE KAART

Translation:

BLOM handed COCU a YELLOW CARD.

VOS received a RED card. / ?? VOS received a RED CARD.

2.6 Speech generation in D2S

The D2S system currently has two different output modes available in the SGM. One is phonetics-to-speech synthesis and the other is phrase concatenation. These modes are discussed in more detail below.

2.6.1 Phonetics-to-speech

Phonetics-to-speech generates speech not from unrestricted text, as in text-to-speech, but from a phonetic transcription with prosodic annotations. This means that the linguistic analysis stage in text-to-speech is skipped. Because the LGM generates an orthographic representation with a unique phonetic representation, it is possible to do errorless grapheme-to-phoneme conversion by lexical lookup instead of rules. The speech output is generated by concatenating diphones, small speech segments consisting of the transition between two adjacent phonemes. A complete diphone inventory for a language covers all possible transitions between any two sounds of that language. The phonetics-to-speech system Calipso, developed at IPO, uses a method called *phase synthesis*, which combines the advantages of PSOLA and mixed-excitation LPC to achieve an output quality that is quite high (Gigi and Vogten 1997, Klabbers 2000). The intonation rules in this system are based on the IPO Grammar of Intonation ('t Hart et al. 1990), which describes the intonation of a sentence in terms of pitch movements. It assigns (combinations of) pitch movements on the basis of combinations of accents and boundaries. In two anonymous tests concerning subjective evaluation under telephone conditions, Calipso was judged favourably on several aspects, including general quality, intelligibility and voice pleasantness (Rietveld et al. 1997, Sluijter et al. 1998).

Current diphone synthesis systems reach a high degree of intelligibility. However, recent evaluations show that when synthetic speech is sent through a telephone channel, intelligibility decreases significantly. In GSM (mobile phone) conditions, intelligibility drops even further (Rietveld et al. 1997). Furthermore, the naturalness of diphone synthesis still leaves a great deal to be desired. Still, it was implemented in the D2S system because it offers unlimited flexibility. In addition, it allows for the testing of the prosody assignment algorithm used in the system, because the prosodic realisation of synthesised speech can be controlled to a large extent.

Specific aspects of the diphone synthesis system which need to be improved in order to achieve more natural sounding speech output, are duration control and the occurrence of audible discontinuities at diphone boundaries. These issues are discussed in detail in Klabbers 2000.

2.6.2 Phrase concatenation using prosodic variants

Unlike speech synthesis, phrase concatenation offers a speech quality that is close to that of natural speech. IPO's advanced phrase concatenation technique, developed by Klabbers 2000, can be seen as a 'prosodically sophisticated' version of the simple concatenation technique discussed in Chapter 1. The most important difference is that in the IPO approach, the phrases are recorded in different prosodic versions, which results in a speech quality that approaches that of natural speech.

The use of several prosodic variants relates especially to the slots in the templates. The fixed parts of the syntactic templates can usually be recorded as a whole, and in only one version. The prosody of the slots in the templates however, is most crucial, because there the variable (and usually most important) information is inserted. These slot fillers were recorded in six prosodically different versions, one for each context described in terms of prosodic markers (i.e., accent markers and phrase boundary markers). Stylisations of these prosodic realisations are depicted in Figure 2.15 and are explained below.


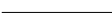
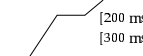
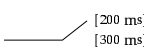
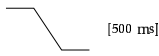
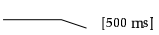
Accent Boundary	Yes	No
None	(1) 	(4) 
Minor / Major continuation	(2) 	(5) 
Finality	(3) 	(6) 

FIGURE 2.15: Stylised examples of the different prosodic versions that are used. Two factors determine their pitch and pausing: the accentuation and the position relative to a minor/major/final phrase boundary. The pauses are indicated between brackets.

1. An accented slot filler which does not occur before a phrase boundary is produced with the most frequently used pitch configuration, the so-called (pointed) *hat pattern*, which consists of a rise and fall on the same syllable. This contour corresponds to the prosodically neutral version used in many other phrase concatenation techniques.
2. An accented slot filler which occurs before a minor or a major phrase boundary is most often produced with a rise to mark the accent and an additional continuation rise to signal that there is a non-final boundary. A short pause follows the constituent, which is 200 ms in length in case of a minor boundary (/) and 300 ms in case of a major boundary (/).
3. An accented slot filler which occurs in final position receives a final fall. It is followed by a longer pause of 500 ms.
4. Unaccented slot fillers are pronounced on the declination line without any pitch movement associated with them.
5. Unaccented slot fillers occurring before a minor or a major phrase boundary only receive a small continuation rise. This prosodic situation does not occur very often. The LGM usually puts a minor or major phrase boundary immediately after an accented word. Again, a 200-ms or 300-ms pause is inserted.
6. Unaccented slot fillers in a final position are produced with final lowering, i.e., a declination slope that is steeper than in other parts of the utterance. They are followed by a 500-ms pause.

When recording the material for the phrase database, the slot fillers, such as player names and time expressions, were embedded in dummy sentences that provide the right prosodic context. The sentences were constructed in such a way as to make the speaker produce the right prosodic realisation naturally. The speaker received no specific instructions about how to produce the sentences.

To concatenate the proper words and phrases during generation, an algorithm has been designed that performs a mapping between the enriched text (i.e., text with accentuation and phrasing markers, as provided by the LGM) and the pre-recorded phrases that have to be selected. The different prosodic variants are chosen on the basis of the prosodic markers. The algorithm recursively looks for the largest phrases to concatenate into sentences. It works from left to right. First, it tries to find the string of N words that contains the entire sentence. If it is present, it is retrieved and can be played. If not, the string comprising the first $N - 1$ words is looked up. This process continues until a match-

ing phrase is found. Then the remaining part of the sentence undergoes the same procedure, until the entire sentence can be played.

The IPO phrase concatenation method has been evaluated in a formal listening experiment (Klabbers 2000), in which it was compared to (i) natural speech output, (ii) a conventional concatenation approach (as often used in commercial applications), and (iii) IPO's diphone synthesis. The results show that the IPO phrase concatenation compares well to natural speech on both intelligibility and fluency, and scores very well on overall quality and suitability. The conventional concatenation approach scores significantly less on all dimensions than IPO's phrase concatenation, indicating that it sounds less natural than is sometimes assumed (Sluijter et al. 1998). The evaluation results indicate that it is worth the extra effort to take a prosodically sophisticated approach to phrase concatenation. Diphone synthesis scores worst on all dimensions. However, in applications where the vocabulary is large, or changes frequently, phrase concatenation is infeasible and speech synthesis is the only option available.

2.7 Discussion

In the generic data-to-speech system D2S there is a tight coupling between the language generation and the speech generation modules. Language and speech generation in D2S is done by means of techniques which incorporate linguistic insights while achieving practical usability. Language generation is done using a hybrid technique where the use of syntactically enriched templates is constrained by local conditions on the linguistic context, while for speech generation pre-recorded phrases are combined in a sophisticated manner, taking prosodic variations into account. Speech generation can also be achieved by phonetics-to-speech synthesis, which offers greater flexibility but a less natural speech quality. The coupling between the language and speech generation modules is brought about by the computation of prosody by the LGM. This ensures that the syntactic, semantic and discourse knowledge captured in the LGM can be used in speech generation, without having the SGM compute the required information anew. Since linguistic analysis of the generated texts would provide less reliable information than is available from the LGM, the IPO approach also makes it possible to achieve a better prosodic quality of the system's output than could be obtained by simply feeding the outcome of language generation into a text-to-speech system.

D2S is a practically useful system which can serve as a basis for a wide range of applications. Systems developed on the basis of D2S can

be run efficiently on PC/Windows and on Unix platforms. Porting D2S to a new application mainly involves constructing a set of syntactic templates, designing a structure representing the input data and, optionally, adding a domain database; all other parts of the LGM are application independent. The work to be done on speech generation depends on the chosen output mode. If phonetics-to-speech is chosen, an off-the-shelf speech synthesis program may be used and only an application-specific lexicon for grapheme-to-phoneme conversion has to be made.⁹ The use of phrase concatenation gives rise to more work, but this is compensated by a more natural sounding speech output (Klabbers 2000).

Although D2S is presented in this chapter as one integrated system, the techniques described here can also be used independently. A variant of the phrase concatenation method used in D2S has been employed in a new version of the German train information system described in Aust et al. 1995, while the LGM of D2S has been used for the generation of English and German route descriptions in VODIS, a European project aimed at the development of a speech interface for a car navigation system (Krahmer et al. 1997, Pouteau and Arévalo 1998).

A major advantage of the use of syntactic templates in the LGM of D2S is that there are no restrictions on the complexity of the sentences that can be generated, nor on the type of information that can be expressed. The variation in the generated texts, achieved by the use of local conditions on the templates, is an important feature of the system, especially in the light of applications like DYD and GoalGetter, where the user is likely to hear a number of generated texts in succession. At the same time, the focus on variation makes D2S somewhat less suitable for some other types of applications. For instance, for feedback messages and prompts in dialogue systems, variation does not seem beneficial, since users may try to interpret variation as meaningful even though it is not. An example is the generation of prompts in the OVIS spoken dialogue system, where D2S is used for spoken output generation. The advantages and disadvantages of using the LGM for language generation in OVIS are discussed in Chapter 5.

In addition, there are some restrictions on the *domain* of potential D2S applications, due to the use of handmade templates in the LGM and of pre-recorded phrases in the SGM (in phrase concatenation mode). This approach is only realistic in domains that are limited in size and subject to few changes. In a very large domain (e.g., general medicine),

⁹Alternatively, an automatic grapheme-to-phoneme converter can be used; currently, such systems achieve an error-rate of approximately 2% (van den Bosch 1997). For names, however, hand-made lexicon entries are still required.

development of a new D2S application would be very time-consuming, and in a very changeable domain (e.g., current theatre or film performances), maintenance might become a problem.

Finally, there are specific aspects of language and speech generation in D2S that need to be improved. In Chapters 3 and 4 of this thesis, some improvements to the LGM and its prosody component are discussed; for a discussion of segmental and prosodic improvements to speech generation in D2S, see Klabbers 2000. The improvements to the LGM proposed in this thesis are in the following areas: (i) the automatic detection of contrast (Chapter 3) and (ii) the generation of referring expressions (Chapter 4). The original prosodic component of the LGM, as discussed in the current chapter, lacks a method for detecting contrastive information in constructions that are only contrastive in a specific context. This sometimes leads to unwanted deaccentuation of phrases expressing contrastive information. In Chapter 3, it is shown that a principled way of determining such ‘contextual contrast’ is required, and a practical, but theoretically motivated way of detecting the presence of contrastive information within the LGM is presented. Next, in Chapter 4, the generation of referring expressions is discussed, with a focus on *distinguishing definite descriptions*, i.e., definite descriptions that can be used to identify a unique referent. The LGM lacks a general procedure for determining which properties to include in such a definite description. Chapter 4 presents an application- and domain-independent algorithm for the context-sensitive generation of definite descriptions, which is based on the well-known Dale and Reiter algorithm for definite descriptions (Dale and Reiter 1995).

Contrastive accent

3.1 Introduction

This chapter focuses on *contrast* as an important factor for accent assignment in spoken language generation. It is generally accepted that in Dutch, as in other Germanic languages, accent functions (among other things) as a marker of *information status*:¹ simplifying, words or phrases are usually accented if they express information that is assumed to be *new* to the hearer, whereas they are usually unaccented if they express *given* information – i.e., information that is assumed to be known to the hearer, for instance because it is available from the preceding discourse. Additionally, words expressing *contrastive* information are always accented, even if this information may be regarded as given. In this chapter, the term ‘contrastive accent’ is used to refer to accents that are used to mark contrastive information. The issue whether there is a phonological difference between contrastive accents and accents that are used to mark new information is not addressed here; in D2S it is assumed there is no difference between the two. (See Krahmer and Swerts 1998 for some discussion on this controversial issue.)

In order to generate acceptable accentuation patterns, a spoken language generation system should be able to determine the information status (new or given, contrastive or not contrastive) of the words and phrases in its output. So far, most research on information status has focused on the distinction between given and new information (see e.g., Prince 1981, Bock and Mazzella 1983, Brown 1983, Terken and Nootboom 1987, Terken and Hirschberg 1994, Birner 1994). Contrast of information has received less attention, possibly due to the elusiveness of the notion. Cruttenden 1986:90: “*Contrastive [...] is a term which most intonationists have to use but for which they find it difficult to give*

¹See Section 2.5.2.

any precise definition.” As a consequence, although the original LGM (as discussed in Chapter 2) contained a fairly sophisticated algorithm for determining which information is new or given, it did not incorporate a principled way of determining contrast. Only fairly recently, some formal theories of contrast have been proposed, a few of which deal with the prediction (or recognition) of contrast. These are discussed in the current chapter. Since these theories could not be straightforwardly implemented in the original LGM, an alternative, more practical method for detecting contrast in the LGM was developed.

The outline of the chapter is as follows. Section 3.2 discusses some of the limitations of the LGM’s original method of prosody assignment, showing that a principled way of determining contrast is required. In Section 3.3 some early, informal views on contrast are discussed, followed by a discussion of some more recent, formal approaches to the prediction of contrast in Section 3.4. Section 3.5 describes a small experiment that was carried out to test which contexts trigger a preference for contrastive accent. Finally, Section 3.6 presents a practical, but theoretically motivated way of detecting the presence of contrastive information within the LGM.

Parts of this chapter are based on earlier publications: earlier versions of Section 3.4 and Section 3.6 appeared in Theune 1997a;1997b, and Section 3.5 is an extended version of Theune 1999.

3.2 The need for contrastive accent

As discussed in Chapter 2, in the LGM accent is assigned only to phrases which are marked as being *in focus*. A phrase is regarded as being in focus when at least one of its words expresses new or contrastive information. The LGM’s mechanism for determining whether a phrase expresses *new* (i.e., *not given*) information is theoretically motivated and rather elaborate (for a description, see Section 2.5.2). On the other hand, the original LGM had no way of automatically determining *contrast*: contrastive phrases were simply marked by hand in the syntactic templates. This means that a phrase could only receive contrastive accent if during the development of the application it was foreseen that this phrase would be used contrastively, for instance being part of a syntactic template expressing an explicit contrast. An example of such a template is Template 1, which is a representation of a template from the DYD system (slightly adapted from van Deemter 1998).

TEMPLATE 1: +C<composition> was also composed <place>, but this was not until +C<time>.

This template is used to express information about a composition that was written in the same place as another, previously mentioned, composition, but at a later date. In the template, the slots for the composition and the date are marked as contrastive, indicated by +C. If a sentence is generated from this template, contrastive accent is assigned to the phrases filling these marked slots.

Template 1 is an example of a template that is specifically designed for expressing a contrast between compositions. However, there are syntactic templates that are in themselves neutral, but which may still express a contrast if used in a certain context. An example is Template 2, which corresponds to a neutral version of the first clause of Template 1.

TEMPLATE 2: <composition> was composed <place>

In (1), Template 2 is used in a context that is neutral with respect to contrast, and in (2) and (3) it is used in a context where intuitively, the composition “*La Chasse*” and the place *Paris* are contrasted respectively with “*Jeunehomme*” and *Salzburg* mentioned in the previous sentence. These contextual differences are reflected in the accentuation patterns that should be assigned to the final sentences in these examples: in (1), *there* is deaccented due to givenness and, due to the application of the Default Accent rule (see Section 2.5.3), the accent shifts to the verb *composed*, whereas in (2) and (3) *Paris* should be accented because it is contrastive.

- (1) In 1777, Mozart lived in Paris₁. “LA CHASSE” was COMPOSED there₁.
- (2) Mozart wrote “Jeunehomme” in Salzburg. “LA CHASSE” was composed in PARIS.
- (3) Around 1778, Mozart₁ left Salzburg and travelled to Paris₂. He₁ wrote “Jeunehomme” in Salzburg. “LA CHASSE” was composed in PARIS₂.

The contrasts in (2) and (3) could not be marked in advance in the template because they arise from the context; therefore they cannot be detected by the original LGM’s accentuation algorithm. (This type of contrast is further referred to as *contextual contrast*.) In example (2) the system’s inability to detect such contextual contrast happens to cause

no accentuation problems: since the contrastive phrases “*La Chasse*” and *Paris* introduce new information to the discourse, the accentuation algorithm will regard them as being in focus and accent them anyway. In example (3) however, an accentuation problem does occur. Here, the original accentuation algorithm will regard *Paris* as being out of focus due to its previous mention (which makes it object-given, as discussed in Section 2.5.2), and deaccent it as a consequence.

In short, the problem with the limited approach to contrast in the LGM is that in cases of contextual contrast (which cannot be specified in the syntactic templates), the LGM’s accentuation algorithm only assigns an accent to the contrastive phrases if they happen to be *new* as well. If the contrastive information is given (or deemed ‘inherently’ unaccentable by the algorithm, for instance in the case of determiners), the algorithm produces the wrong results. In DYD, this never caused any real problems, since all cases of contrast were handled by specific templates like Template 1. ‘Contextual contrast’ simply did not occur. (The above examples were forged to illustrate the possible negative consequences of the system’s lack of a structural treatment of contrast.) In the GoalGetter system, however, cases of contextual contrast occur quite frequently, and in some of those cases the LGM’s original accentuation algorithm failed to accent the contrastive items. A typical example is shown in (4).

- (4) *Feyenoord nam na drie minuten de leiding door een goal van Koeman. In de zestiende minuut verzilverde Kluivert een strafschop voor Ajax. Tien minuten later scoorde LARSSON een goal voor FEYENOORD / ?? Feyenoord.*

Translation:

After three minutes, Feyenoord took the lead through a goal by Koeman. In the sixteenth minute, Kluivert kicked a penalty home for Ajax. Ten minutes later, LARSSON scored for FEYENOORD / ?? Feyenoord.

In the final sentence of (4), the player *Larsson* and the team *Feyenoord* appear to stand in contrast to the player *Kluivert* and the team *Ajax* that were mentioned in the preceding sentence, and it therefore seems most natural to accent both. However, the original accentuation algorithm of the LGM deaccented the final *Feyenoord* due to givenness, because *Feyenoord* was already mentioned in the first sentence of the text. The resulting accentuation pattern does not seem fitting, since it does not adequately reflect the contrast between the two sentences.

(As we will see, the subjects in the perception experiment described in Section 3.5 shared this intuition for similar constructions.)

Rump and Collier 1996 have shown for Dutch that listeners are very well able to recognise the focus structure of an utterance on the basis of its accentuation pattern alone. This means that in data-to-speech, it is very important to generate an accentuation pattern which matches the hearer's expectations concerning the focus structure of the generated utterance, i.e., which items are in focus due to newness or contrast. A clashing accentuation pattern is likely to sound unnatural and confuse the hearer. Therefore, incorrect deaccentuation of items that should be in focus due to contrast should be avoided.

So far, the assumption has been that improper deaccentuation by the LGM in cases like (3) and (4) is caused by a failure to recognise contextual contrast. The remainder of this chapter focuses on determining the contextual factors that cause certain phrases to be perceived as contrastive, and how these can be automatically detected in the LGM. But first, let us briefly consider an alternative explanation for the required accentuation of the given items *Paris* and *Feyenoord* in the final sentences of (3) and (4). This explanation for their accentuation may be derived from a perception experiment by van Donzel 1999, in which listeners judged the prosodic prominence of items in different 're-told' or 're-read' versions of a short story (van Donzel 1999:28-35). The items were classified in terms of information status, based on the taxonomy of Prince 1981. Following Brown 1983, van Donzel subdivided Prince's class of 'textually evoked' entities – i.e., items that are given because they were mentioned in the preceding discourse – into a 'current' and a 'displaced' class. A textually evoked, displaced item has been mentioned in the discourse, but is no longer the most salient discourse entity, and as a consequence it cannot be referred to using a pronoun. Van Donzel found that displaced entities were relatively often perceived as prominent, and conjectured that such entities "have to be 'refreshed' (as a resumed topic) and thus are 'new' in a certain sense" (van Donzel 1999:30). This effect has already been noted by Chafe 1976, who observes that a previously mentioned item which is no longer in the listener's consciousness, nor recoverable from the preceding discourse (cf. Halliday 1967), no longer counts as given. The results of the experiment described in Section 3.5 also seem to support this finding. (See Section 3.5.6.)

Turning back to the second occurrences of *Paris* and *Feyenoord* in examples (3) and (4), we see that both expressions refer to 'displaced' entities: *Paris* and *Feyenoord* are not the most recently evoked entities in (3) and (4) respectively, and pronominalisation is impossible in both cases. So, an alternative explanation (other than contrast) for the accen-

tuation of these two expressions is that their referents have become less accessible when other items were introduced in the discourse, and that the attention of the hearer must be redirected to them using (among others) prosodic means. Since the LGM's accentuation algorithm only takes paragraph boundaries into account, and does not further distinguish 'displaced' from 'currently evoked' items, it simply deaccents all subsequent references that fall within the same paragraph, regardless of the distance to their antecedent.

Another, somewhat similar, alternative explanation for the accentuation of previously mentioned items in examples (3) and (4) is that the 'contrastive' items in these examples are part of a new discourse segment, in which the entities need to be re-introduced. Indeed, in (3) there is a shift from Salzburg to Paris, and in (4) there is a shift from one team to another. However, these shifts do not seem to be related to the start of a new discourse segment, as the examples under discussion are very short. Therefore, any reduction in givenness of the entities mentioned in these texts appears to be related to local (intra-segmental) rather than global (inter-segmental) discourse structure.

What both alternative explanations discussed above have in common, is their implication that the LGM's accentuation problems with examples like (3) and (4) are caused by not setting the proper limits for givenness. However, although such an explanation may account for the examples discussed so far, it fails to explain other cases where given items are accented. This can be illustrated by the following example from Lakoff, who observes that in a construction like (5), the pronouns must be accented as indicated:

- (5) John insulted Mary and then SHE insulted HIM.
(Lakoff 1971:333)

In (5), the accented items cannot be said to be 'displaced', as both items are fully accessible and pronominalisation is perfectly acceptable. In general, pronominalisation is only felicitous if the antecedent is somehow accessible (either from the preceding discourse or from the situational context) and it can therefore be argued that pronouns must always refer to given information. Still, accentuation of pronouns is very well possible. This is not only shown by (constructed) examples like (5), but also by the occurrence of accented pronouns in real-life conversations: Prevost 1995 found that in constructions of the form *but he ...*, occurring in the Switchboard corpus, 19.4 % of the subject pronouns was accented. In short, examples of contrast involving pronouns, like (5) or the "*but he*" cases from the Switchboard corpus, show that contrast must be an independent trigger of accentuation. In the other

examples discussed in this section, which do not involve pronouns, the accentuation of previously mentioned items is also assumed to be caused by contrast. This is not to say that contrast is the *only* factor attributing to the accentuation of previously mentioned items; however, it clearly is an important one that must be taken into account in data-to-speech generation. In order to do this successfully, a formal (and potentially computational) treatment of this intuitive notion is required.

3.3 Early views on contrast

In the early, informal discussions of contrast available in the literature, contrast is often described in terms of ‘sets of alternatives’ and/or ‘parallelism’, terms which reoccur in most of the recent, more formal approaches. The meaning of these terms has never been quite clear and they have certainly not been interpreted in the same way by everyone. For example, Chomsky 1971 notes that “in ‘parallel constructions’, in some sense of this notion that has never been made quite clear, contrastive intonation is necessary” (Chomsky 1971:205). He then goes on to argue that it is a parallelism of surface structure, not of deep structure, that plays a role here, using example (6) as an illustration.

- (6) John is neither EASY to please, nor EAGER to please, nor CERTAIN to please (...)
(Chomsky 1971:205)

Ladd 1980 opposes this view and claims that parallelism is not a syntactic, but a semantic notion. Using (7) as an example, he shows that it is not syntax which triggers a specific accentuation pattern, but that it is the speaker who chooses to accent certain items to set up a semantic parallelism of his choice.

- (7) A: When was the last time you saw any of your relatives?
B: My mother called me yesterday – does that count?
(Ladd 1980:79)

Here, according to Ladd, the speaker could choose to set up a parallelism between ‘(relative) (get in touch with) (member of conversation)’ and ‘(member of conversation) (get in touch with) (relative)’ by accenting *mother* and *me*. Alternatively, he could accent *called*, highlighting it as the focus of the utterance and as the antecedent for *that*, thereby comparing it with *seeing*. Ladd thus shows that what counts is not syntax, but the speaker’s intention of setting up a parallelism between certain items, or combinations of items. Similarly, according to Bolinger (referred to in Lakoff 1971) accentuation of the pronouns in example (5) is only required if it is the individuals involved that are being contrasted,

whereas if the speaker intended to contrast the time of the events, the accentuation would be as follows:

- (8) John insulted Mary and THEN she insulted him.
(Lakoff 1971:333)

Ladd and Bolinger suggest that a speaker can choose which parts of an utterance to accent, in order to set up a parallelism or contrast with parts of a preceding utterance. Ladd 1980:80: “*if, and only if, the speaker wants the accent pattern that says ‘parallel’, he uses it.*” Ladd does not make clear, however, what it is that makes certain parts of an utterance eligible for the accenting that signals parallelism. In (6), these parts happen to be “the points of difference in otherwise identical phrases” (Ladd 1980:79), but this need not always be the case, as example (7) shows.

Still, it appears that in many cases accenting is not optional: for instance, in examples (3) and (4) the speaker has no choice but to accent “*La Chasse*” and *Paris*, and *Larsson* and *Feyenoord* in the final sentences. All other accentuation patterns seem unacceptable. Here, accentuation seems to be forced on, rather than chosen by, the speaker. Even in example (8), it seems quite unnatural *not* to accent the pronouns *she* and *him*. Although the speaker appears to have some free choice in whether or not to contrast the times of the events, such a free choice seems to be absent in the case of the pronouns. Apparently, the reversal of the roles of John and Mary inevitably triggers a sense of contrast. According to Schmerling 1976, it is the “newness of the semantic relations” which causes the pronouns in (5) to be accented. As a further illustration, she discusses example (9) from Akmajian and Jackendoff 1970.

- (9) John₁ hit Bill₂ and then George hit {him₂ / HIM₁}.

If the pronoun *him* in the second conjunct of (9) refers to Bill, this is not a new semantic relation (since Bill was hit before) and the pronoun is not accented; however, if *him* refers to John, this *is* a new semantic relation and the pronoun is accented in order to draw attention to this fact.

Other authors have attempted to explain why speakers use contrastive accent and which utterance parts are eligible for receiving it in terms of ‘sets of alternatives’, without referring to parallelism. For instance, Chafe states that the function of a contrastive sentence is to point out which of a limited number of candidates for a certain ‘role’ is the correct one (Chafe 1976:34); Cruttenden informally defines contrast as “involving comparison within a limited set” (Cruttenden 1986:90); and Bolinger says that in cases of contrast “one or more individual

items are singled out from a larger (but limited) set as being true as regards some relationship whereas others in the same set are untrue" (Bolinger 1986:91). They all agree that in one sentence, more than one item can be contrastive; e.g., in (2) and (3), "*Jeunehomme*" is contrasted with "*La Chasse*", and *Salzburg* with *Paris*.

An important point of discussion is what actually counts as a 'limited set'. As Cruttenden observes, the clearest cases of contrast involve an explicit comparison between pairs of items (binary sets); and such examples are most often cited as examples of contrast. All the examples discussed in Section 3.2 are of this type, involving binary sets: {"*Jeunehomme*", "*La Chasse*"}, and {*Salzburg*, *Paris*} in examples (2) and (3); {*Kluivert*, *Larsson*}² and {*Ajax*, *Feyenoord*} in example (4); and {*John*, *Mary*} in example (5). In example (6), a number of adjectives (*easy*, *eager*, *certain* ...) are contrasted. Ladd's example (7) is less easily explained in terms of limited sets of alternatives. The same goes for many other examples quoted in the literature, especially those involving *implicit* contrast, i.e., cases of contrast where the items with which the contrastive item is compared remain unmentioned. In those cases it is not easy to determine the content (and therefore size) of the set of alternatives. For instance, in the following example *it* seems to be contrastive, but it is unclear with what it is contrasted.

- (10) The TUC is launching a fifty thousand pound CAMPAIGN / to counter what IT calls / the Government's anti-union POLICY.
(Cruttenden 1986:90)

For cases of implicit contrast involving closed categories with a relatively small number of elements (e.g., colour names, spatial prepositions), it is easier to determine the set of alternatives, but for such cases it is often unclear whether newness or contrast is involved. For instance, in the example below, *red* obviously constitutes new information, but it is not easy to determine whether it is *contrastive* as well.

- (11) John was trying on different t-shirts, and finally put on a RED one.

Here, due to deaccentuation of the anaphor *one*, the accent in the NP *red one* falls squarely on *red*, thus increasing the sense of contrast. This effect is referred to by Ladd 1980, who points out that many cases of so-called 'contrastive stress' are actually "a by-product of deaccenting which I will call **default accent**" (Ladd 1980:81, see also Section 2.5.3). Indeed, the following variant of example (11), in which there is

²Possibly, the first mentioned player *Koeman* should also be added to this set of alternatives.

no deaccentuation within the final NP, does not give quite such a strong impression of contrast:

- (12) John was trying on different clothes, and finally put on a RED T-SHIRT.

Clearly, it is often very difficult to draw the line between newness and contrast. According to some, such a line does not even exist, because contrast is merely a variant of newness, where the main difference between the two lies in the number of possible alternatives (see e.g., Halliday 1967, and more recently Piwek 1998). Here, this point is not discussed any further; contrast is simply assumed to be inherently different from newness. Moreover, this chapter focuses on *contextual contrast*, which is always ‘explicit’ in the sense that the accented items are contrasted with something that was mentioned earlier in the discourse.

3.4 Formal approaches to contrastive accent assignment

In Section 3.3 we have seen that most intonation experts use intuitive, informal notions like parallelism or limited sets. The current section offers a brief discussion of a few relatively recent approaches to the prediction of contrast that aim at making these notions more explicit. The discussion will be restricted to examples involving contrast between two subsequent sentences. The approaches to contrast discussed here are those proposed by Prevost, Pulman (and some others working in the HOU framework) and van Deemter.

3.4.1 Prevost: sets of alternatives

The theory of contrast proposed by Prevost 1995 was inspired by the ‘alternative semantics’ of Rooth 1985;1992.³ Prevost’s approach to recognising contrast is mainly based on determining if ‘alternative items’ have been mentioned in the previous discourse. Prevost also discusses the determination of other aspects of information structure, as well as content planning and the generation of surface form, including intonation markers. These issues are not discussed here.

In Prevost’s approach, contrast is determined at the level of semantic representation, where two propositions count as contrastive either when they contain two contrasting pairs of discourse entities or when they contain only one contrasting pair of discourse entities plus contrasting ‘functors’, e.g., verbs. (This appears to be a form of checking for *parallelism*, although Prevost does not use this term.) A discourse entity *x* is

³Although Rooth deals with contrastive accent as well, his theory is not discussed here because it is purely aimed at the *interpretation* of focus (including contrastive accent), not its prediction.

contrastive if the preceding discourse contained a reference to another entity belonging to the ‘set of alternatives’ of x , which is a set of different entities of the same type as x . Of contrastive *functors*, no definition is given, but the verbs *love* and *hate* are given as an example.

Prevost implemented his theory in two small generation systems, one of which can produce the responses to database queries, while the other produces monologues describing stereo systems and their components. A sequence of two contrastive utterances, generated by the monologue system is shown in (13).⁴

- (13) The x4 is a SOLID-state AMPLIFIER. The x5 is a TUBE amplifier.
(Prevost 1995:142-143)

The generation of the second utterance, in the context of the first, is discussed in some detail by Prevost. In the utterance, two discourse entities are referred to: **e2**, an amplifier (expressed using the name $x5$), and **c2**, which is a specific *class* of amplifiers (expressed using the description *a tube amplifier*). These two entities are considered to stand in contrastive relationships to the entities referred to in the first utterance: **e1**, a different amplifier, and **c1**, which is a different *class* of amplifiers. This means that the requirement of two contrasting pairs of discourse entities is met. Once the contrastive entities are located, Prevost’s algorithm for contrastive accent assignment determines which of these objects’ properties need to be contrasted. This is done as follows. First, the algorithm determines the set of all alternatives to the contrastive object x that were explicitly mentioned in the prior discourse. Together with x itself, this set forms the initial *RSet*. Then, for each property of x in turn, the *RSet* is restricted to include only the objects satisfying the given property. If this decreases the cardinality of the *RSet*, then the property under consideration is regarded as contrastive, since it serves to distinguish x from its alternatives.⁵ In this way, it is determined that **c2**’s having a *tube* design is a contrastive property, but its being an amplifier is not. Consequently, the modifier *tube* receives contrastive accent, and the noun *amplifier* does not.

The main problem of Prevost’s approach, as Prevost himself notes, is that the problem of what constitutes a set of alternatives (see Section 3.3) is still unsolved, as it is difficult to define exactly which items count as being of the same type. If the definition of ‘same type’ is too strict,

⁴In the example output presented by Prevost, these utterances are not adjacent. They are the first sentences of two subsequent paragraphs describing the two amplifiers. The example is shortened here for expository reasons.

⁵This procedure is similar to the property selection in Dale & Reiter’s Incremental Algorithm for the generation of referring expressions, discussed in Chapter 4.

not all cases of contrast will be accounted for. On the other hand, if it is too broad, then anything will be predicted to contrast with anything. Prevost discusses the following problematic example:

- (14) Mary took John to a hockey game for his birthday, but he didn't seem very pleased. While HE intently watched the CLOCK, SHE watched the GAME.
(Prevost 1995:147)

This is a clear case of contrast, but it does not seem appropriate to regard *clock* and *game* as being in the same set of alternatives, since they are not obviously of the same type. Counting them as alternatives would mean an unwanted broadening of the notion of 'set of alternatives' (such a set could then contain almost anything, and could hardly count as limited). Prevost has adopted a practical solution in the spoken language generation system he developed: in this system, alternative sets are fixed in the knowledge base, where such sets are inferred from **isa** links that define class hierarchies (Prevost 1996). Only entities with the same parent or grandparent class are considered alternatives. Thus, the system is unable to produce contrastive accent in examples like (14). A second problem with Prevost's approach is that there are cases where his condition on contrastive propositions seems to be too strict. Consider:

- (15) BLIND scored a goal for AJAX. KLUIVERT scored a goal TOO.

Examples like (15) are generally considered contrastive (see also Section 3.4.3), but they fall outside the scope of Prevost's algorithm, because that requires the presence of *two* pairs of contrastive entities, or one pair and a pair of contrastive functors. In this example, there is only one pair of contrastive entities, i.e., the two football players Blind and Kluyvert, and the verbs are not contrastive. This means that in Prevost's system, *Kluyvert* in (15) is not eligible for contrastive accent. A possible way out of this problem can be based on the following claim by Kamp 1996:

Contrast requires a minimum of **two** focused elements. When the asserted proposition differs in only one constituent from the context proposition with which it is to be contrasted, this requirement must nevertheless be maintained. Usually this is done by adding a stressed element that expresses the relation in which the new proposition stands to the context proposition, thereby indicating some additional respect in which the content of the new utterance differs from the context. (Kamp 1996)

In the case of example (15), Kamp would say that the required second focused element is *too*. Prevost could incorporate this explanation in his algorithm by considering *too* and similar words as expressions for contrastive functors in the semantic structure. Still, although Kamp's observation seems to be correct for examples like (15), it is not a priori clear that a contrastive proposition should always contain two contrastive items. In the answer of (16), clearly contrastive accent is required on British, although no other contrastive entities than the British amplifier are mentioned, and there is no contrastive functor. This remains unexplained in Prevost's theory.⁶

(16) Q: Does the BRITISH or the AMERICAN amplifier produce clean treble?

A: The BRITISH amplifier produces clean treble.

(Prevost 1995:81)

3.4.2 Pulman and others: HOU and parallelism

Another approach to the generation of contrastive accent is advocated by Pulman 1997, who, like Ladd 1980, relates contrast to parallelism. Pulman proposes to use Higher Order Unification (HOU) for the interpretation and prediction of contrastive accent. Higher Order Unification is a method for solving equations through substitution, where the sides of the equations are terms of higher order logic, i.e., a logic that allows variables of any type, not just variables ranging over individual entities as in first order logic. Pulman makes use of equivalences like the one in (17), which can be used for both interpretation and prediction of contrast, and which operate at the level of semantic representation (or, more specifically, quasi-logical form or QLF, see Alshawi and Crouch 1992).

(17) $\text{assert}(F, S) \Leftrightarrow S$
 if
 $B(F) = S$
 $\text{context}(C)$
 $P(A) = C$
 $\text{parallel}(B \bullet F, P \bullet A)$

In (17), S corresponds to the QLF of the target utterance (i.e., the utterance for which the focused parts must be found); F corresponds to the focused part of S ; B is the background part of S (i.e., the result of abstracting over F in S); C is the QLF of a 'salient' utterance in the

⁶Kamp claims that contrast differs from the question-answer relation (Kamp 1996), and would therefore not consider (16) as a case of contrast. However, similar examples can be found in non-question-answer constructions, e.g., *We have an American amplifier and a British amplifier on sale. I recommend the BRITISH amplifier.*

context of the target utterance; and P and A are the parts of C that are parallel to B and F respectively. Pulman does not define exactly when two items count as parallel, but states that “to be parallel, two items need to be at least of the same type and have the same sortal properties” (Pulman 1997:90). This is similar to (but much less specific than) Prevost’s characterisation of the elements of alternative sets, discussed in Section 3.4.1.

Informally, what the equivalence in (17) says is that asserting QLF S with focus on F is equivalent to S if in its context an utterance can be found with QLF C, where C contains an item A that is parallel to F, while the background P of C is parallel to the background B of S. Using HOU, this equivalence can be resolved in order to predict the landing place of focus markers in a generated utterance. As an illustration, assume that the focus of the second utterance in (18) below must be computed.

- (18) A: John kissed Sue.
 B: (No,) John kissed Mary.

Given the utterances in (18), we know that S is equal to $kiss(j,m)$, being the semantic representation of the second utterance, and that C is equal to $kiss(j,s)$, the semantic representation of the preceding utterance. In order to determine F, the equation in (19) needs to be resolved.

- (19) $B(F) = S = kiss(j,m)$
 $P(A) = C = kiss(j,s)$
 where
 $parallel(B \bullet F, P \bullet A)$

Using HOU, the following solution can be found: $F = \lambda P.P(m)$ and $A = \lambda Q.Q(s)$, with $B = P = \lambda O.O(\lambda x.kiss(j,x))$. Informally, this means that Mary is the focus (i.e., contrastive element) of the second utterance, and that the background of both utterances is that John kissed someone. Note that Pulman allows parallel elements to be identical, as are B and P in this example. When the ‘parallel’ operator takes several arguments, as in (17), at least one parallel pair is required to be distinct; the other(s) may be identical. This allows Pulman to deal with cases like (18), which Prevost cannot handle since they involve only one pair of contrastive (or focused) items. Still, the other problem with Prevost’s approach, namely that of defining when two items are “of the same type”, is also a problem in Pulman’s theory. In fact, Pulman consciously avoids giving a definition of this “notoriously slippery notion” (Pulman 1997:93).

A way of determining which are the parallel elements in parallel constructions based on HOU is proposed by Gardent and Kohlhase 1997.

They combine HOU⁷ with an abductive calculus using sorted type theory (Kohlhase 1994) to model ‘contrastive parallelism’ or in short *c-parallelism*. In sorted type theory, objects of a certain logical type are subdivided in terms of sorts (which can be seen as unary predicates), ordered by a partial ordering relation ($=<$) in a sort hierarchy, as shown in Figure 3.1 for objects of type *e* (entities). Objects are *c-parallel* (and thus belong to each other’s set of alternatives, in Prevost’s terms) if they are both *similar* (have a sort in common) and *contrastive* (have a *distinguishing* (complementary) sort – for instance, one is ‘animate’ whereas the other is ‘inanimate’). There are not only sort hierarchies for objects of type *e* but also for all other types, e.g., relations between entities (*support, oppose, like ...*). Thus, the analysis is not restricted to NPs, which are the main focus of most theories of contrast. The ideas of Gardent and Kohlhase on what it takes for items to be *c-parallel* are largely similar to those of Prevost 1995 concerning alternatives. As a consequence, the approach advocated by Gardent and Kohlhase seems equally unsuitable to deal with examples like (14), which proved problematic for Prevost.

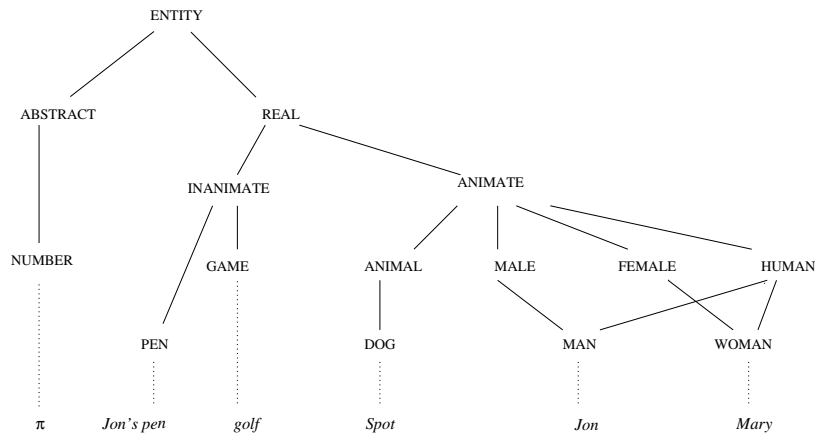


FIGURE 3.1: Sort hierarchy of type *e* (Gardent and Kohlhase 1997).

⁷Actually, they use a variant of HOU called Higher Order Coloured Unification or HOCU, which unlike ordinary HOU does not over-generate (i.e., yield solutions which are linguistically invalid) (Gardent et al. 1998).

3.4.3 Van Deemter: contrariety and equivalence

The theory of contrast put forward in van Deemter 1994a;1998;1999 is based on the determination of contrariety and equivalence. Van Deemter acknowledges that (syntactic) parallelism is a source of contrast, referring to formalisations of parallelism as proposed by Prüst 1992, Hobbs and Kehler 1997 and others, but points out that there are many examples of contrast which lack parallelism. Van Deemter uses the notion of *contrariety* to account for these cases. Informally defined, two sentences (or clauses) are contrary to each other if they cannot be true at the same time. According to van Deemter, two sentences are said to stand in a contrast relationship if they contain two items (one in each sentence) which are ‘contrastible’ and whose substitution by the same constant will cause the two sentences to be contrary to each other. A contrastive relationship between two sentences triggers a contrastive accent in each sentence. The landing places of these accents depends on the location of the substituted items. If they are located in the subject of the sentence, the accent will land in the subject; if they are located in the predicate, the accent will land in the predicate. (The two items need not be located in corresponding parts of the two sentences.) To determine on which word in the subject or predicate the accent will land, a version of Focus Accent theory is used (Dirksen 1992, Dirksen and Quené 1993; see also Chapter 2).

Van Deemter gives (20) as an example. If we assume that being an organ mechanic implies knowing much about organs, as stated in meaning postulate (21), then replacing Mozart by Bach produces a contrariety. This correctly predicts a contrastive accent on Bach and Mozart.

- (20) BACH was an organ mechanic; MOZART knew little about organs.
 After substitution:
 Bach was an organ mechanic; Bach knew little about organs.
 (van Deemter 1999:11)

- (21) $\forall x[\text{organ_mechanic}(x)] \Rightarrow [\text{know_much_about_organs}(x)]$

Two sentences also count as contrastive if they contain two contrastible items which, after replacement by the same constant, cause the sentences to be logically *equivalent* (van Deemter 1999). This is shown in (22). Including equivalence as a source of contrast allows van Deemter to handle cases that are problematic for Prevost 1995 (see Section 3.4.1).

- (22) SEVEN is a prime number and so is THIRTEEN.
 After substitution:
 Seven is a prime number and so is seven.
 (van Deemter 1999:12)

Van Deemter's definition of 'contrastible items' is extremely permissive, the only constraint on contrastibility being inequality of denotations. This permissiveness forces van Deemter to place a restriction on contrastive propositions which is opposite to that of Prevost: contrarieties (or equivalences) which are reached through more than one substitution do not qualify for contrastive stress, because otherwise far too many cases of contrastive stress would be predicted: any pair of sentences of the form $(NP_1 VP_1)$, $(NP_2 \text{ Negation } VP_2)$ or $(NP_1 VP_1)$, $(NP_2 VP_2)$ would then always count as contrastive, since substitution by the same constant of the NPs and of the VPs at the same time would lead to a contrariety or equivalence.⁸

However, many examples of contrast can only be explained in van Deemter's theory if at least two pairs of items are substituted, because substitution of only one pair of items does not lead to a contrariety or an equivalence. For instance, the contrast in Prevost's example (14) (*While HE intently watched the CLOCK, SHE watched the GAME*) could be easily predicted by replacing the pairs {he, she} and {clock, game} with the same constant, which would result in an equivalence. Replacing only one pair of contrastible items does not give the desired results for this example. The best prediction would be made by replacing the pair {he, she} by a constant, because under the assumption that "intently watching the clock" is incompatible with watching a game, this would result in a contrariety (e.g., *While John₁ intently watched the clock, he₁ watched the game*). However, this solution is not completely satisfactory, since it only predicts contrastive accent in the subject part of the contrastive sentences, which is where *he* and *she* are located. It does no justice to the observation that (14) appears to contain *two* contrastive pairs, which are located both in the subject *and* the predicate part of the containing sentences: {he, she} and {clock, game}. Not counting *clock* and *game* as contrastive leaves open the possibility of deaccenting them in a context where they are given.

Another example, based on Prevost 1995, is (23). Here an equivalence can only be reached by substituting constants for the pairs {British, American} and {Stereofool, Audiofad}. Replacement of only one of these pairs does not result in an equivalence or contrariety, as shown below.

- (23) The BRITISH amplifier was praised by STEREOFOL, an audio journal. The AMERICAN amplifier was praised by AUDIOFAD, another audio journal.

⁸This problem is discussed in van Deemter 1994a.

After substitution of two items:

The British amplifier was praised by Stereofool. The British amplifier was praised by Stereofool.

After substitution of only one item:

The British amplifier was praised by Stereofool. The American amplifier was praised by Stereofool.

or:

The British amplifier was praised by Stereofool. The British amplifier was praised by Audiofad.

Of course, the above example and many others of its kind can be easily explained through parallelism, and an explanation in terms of contrariety or equivalence is thus not required. However, in most cases it is possible to come up with re-phrasings of such examples that do not show syntactic parallelism, e.g., (24), and which therefore still lack an explanation. In order to account for such cases of contrast, the restrictions on substituting contrastible items need to be worked out in more detail than is presently the case.

- (24) STEREOFUOL, an audio journal, printed a favourable review of the BRITISH amplifier. The AMERICAN amplifier was praised by AUDIOFAD, another audio journal.

3.4.4 Discussion

As seen in Section 3.3, most of the early, informal definitions of contrast are stated in terms of parallelism or sets of alternatives. These notions return (in one form or another) in later, more formal and/or computationally oriented approaches to the determination of contrast. In the works of Prevost 1995 and Pulman 1997 the two notions are combined. In spite of differences in perspective and the use of computational tools, the general ideas behind the theories of Prevost 1995 and Pulman 1997 appear to be similar: both check the semantic structures of utterances for some kind of parallelism, which for both is strongly linked to the presence of parallel or alternative items. Neither of them provides a method for the automatic detection of those items, but this crucial gap is filled (at least for the HOU-approach advocated by Pulman) by Gardent and Kohlhase 1997. Van Deemter 1994a;1998;1999 also acknowledges the importance of (syntactic) parallelism, but mainly focuses on contrariety and equivalence as sources of contrast. Contrariety and equivalence are detected through substitution of contrastible

items. According to van Deemter, the only condition on contrastibility is inequality of denotations.

Since Prevost and van Deemter place complementary restrictions on the number (> 1 or < 2 respectively) of alternative or contrastible items that a pair of sentences may contain (or that may be substituted in van Deemter's approach), they cannot handle the same examples. Prevost's approach cannot deal with examples like (22), that contain only one pair of alternatives, whereas examples like (23), that require substitution of two contrastible pairs, cannot be explained through contrariety or equivalence (though they can be explained through parallelism). Pulman's approach is suitable to deal with examples of both kinds.

Examples that involve 'implied contrasts', like (20) and (24), are not discussed by Prevost or Pulman. Still, it seems that such cases can be accounted for within their approaches, given that sufficient world knowledge is available (e.g., encoded in meaning postulates like (21)). For instance, (20) can be handled within Prevost's approach, if we assume that "knowing little of" and "being a mechanic of" correspond to contrastive functors on the basis of meaning postulate (21). Similarly, in the HOU approach the contrast between the two clauses of (20) can be predicted by not equating C in equivalence (17) to the direct semantic representation of the first clause, but to its equivalent according to (21). Example (24) might be handled in the same fashion in both approaches, given the availability of the knowledge that "printing a favourable review" is equivalent to "praising". Clearly, the presence of adequate world knowledge is a crucial factor in all accounts discussed here, including van Deemter's.

Finally, examples like (14), which involve items that *in isolation* would never be considered contrastive, seem to pose a problem for the approaches of Prevost and Pulman. Contrastive constructions like (14) are quite common; another example is given below.

(25) Mary came out against John in the finals of a TV quiz. In the end, SHE won a TOASTER; HE won a CRUISE.

Examples like these suggest that any two items may be considered as contrastive *if* they have the same thematic roles (e.g., agent or patient) with respect to the same verb. This observation is in line with Chafe's 1976 analysis of contrast (see Section 3.3). A possible solution for such examples in Prevost's system would therefore be to check the discourse context not only for contrastive functors, but also for functors that are identical (or equivalent) to that of the utterance being generated. If such a functor is present in the context, its arguments should be designated

as contrastive items. Presumably, a similar move could be made within the HOU approach.

To conclude the comparison, we have seen that none of the formal theories of contrast that have been discussed in this section could account for all examples of contextual parallelism that were presented. At first sight, the theory of van Deemter seems to ‘score best’, thanks to the fact that it explains contrast through two different factors: syntactic parallelism and contrariety (or equivalence). Certain cases of contrast that show no syntactic parallelism seemingly require an explanation in terms of contrariety or equivalence. However, as discussed above, with some adaptations to the approaches of Prevost and Pulman it seems possible to explain those examples in terms of parallelism as well.

Despite the differences between the formal approaches discussed here, there is still much they have in common. All of them make use of a notion of alternative items (also called parallel, contrastive, or contrastible items), but the presence of a pair of such items in two utterances (one item per utterance) is generally not considered a sufficient condition for the assignment of contrastive accent. In addition to the required presence of at least one pair of alternative items, all approaches impose a further condition that must be met for two utterances to count as contrastive: Prevost 1995 requires the presence of at least one additional pair of contrastive objects or a pair of contrastive functors; Pulman 1997 requires the ‘backgrounds’ of the two utterances to be parallel; and van Deemter 1998;1999 requires that substitution of the contrastible items in the two utterances results in a contrariety or equivalence.

Generalising over these three approaches, we can say that all of them regard some form of (semantic) *parallelism* as a crucial condition for the assignment of contrastive accent. None, however, provides an empirical validation of this claim. In the next section, a small perception experiment is discussed that was carried out to verify if parallelism is indeed a crucial factor for the assignment of contrastive accent.

3.5 Experiment

This section describes a small experiment that was carried out to test which contexts trigger a preference for contrastive accent. In the experiment, the subjects had to indicate for a number of texts which of two spoken versions of these texts they found most natural-sounding. The two versions differed with respect to the accentuation of one target item.

3.5.1 Hypotheses and assumptions

The following hypotheses were tested in the experiment:

Hypothesis I: the presence of one pair of alternative items in two consecutive sentences triggers a preference for contrastive accent.

Hypothesis II: parallelism between two consecutive sentences triggers a preference for contrastive accent.

Concerning alternative items, parallelism, and the relation between accentuation and information status, the following assumptions are made.

Alternative items are assumed to be similar to the *c-parallel* items of Gardent and Kohlhase 1997, i.e., they are entities (of all logical types) that have both a common and a distinguishing sortal property. Most pairs of alternative items occurring in the experiment are ‘obvious’ alternatives such as {cat, dog}, {car, bike}, {relaxed, nervous}, etc. These are all entities that clearly belong to the same mother category. In order to include ‘less obvious’ alternatives such as *clock* and *game* in Prevost’s example (14), arguments of the same verb are also regarded as alternative items: if two items occur as arguments (in the same thematic role) of the same verb, these are considered as alternatives even if they would not be considered as alternatives in isolation. This extension is in line with the observation from Chafe 1976 that contrast is related to the presence of different candidates for the same role; see the discussion of example (25) in Section 3.4.4. Another example is shown below in (26). Here, the Pope and the ground are considered as alternatives, since they are both being kissed by someone. In (27), however, they are *not* alternatives as they are not used as arguments of the same verb. (Both examples are translations of texts used in the experiment.)

(26) The Pope₁ kissed the ground. The head of state kissed him₁.

(27) The Pope₁ kissed the ground. The head of state congratulated him₁.

The definition of *parallelism* used for the experiment is similar to, but more restrictive than, that of Pulman 1997. It is assumed that two sentences count as parallel if they obey both of the following conditions: (i) they contain at least one pair of alternative items and (ii) their backgrounds (i.e., the results of abstracting over the alternative item(s) in the sentences) are identical (ignoring the presence of modifiers such as *very* and discourse markers such as *then*, *afterwards*, etc.). Thus, the sentences in (26) count as parallel, because they have identical backgrounds of the form “X kissed Y”. The sentences in (27) are not

parallel: although they contain one pair of alternatives (consisting of the Pope and the head of state), their backgrounds are not identical.

The condition that the backgrounds of two contrastive sentences must be *identical* is more restrictive than that of Pulman, who only states that the backgrounds must be ‘parallel’ – a notion which he does not specify, but which includes identity (Pulman 1997:93). The restriction to identical backgrounds was made here for two reasons: first, it takes away the need to define what it takes for backgrounds to be ‘parallel’; second, it restricts parallelism to the clear-cut cases where syntactic parallelism coincides with semantic parallelism, thus excluding examples like (20) and (24).

With regard to *accentuation* and *information status*, the following assumptions are made. Information is either contrastive or not contrastive, and at the same time it is either new or given. (In other words, contrast and given- or newness are orthogonal properties.) Information is given if it has been expressed previously in the same discourse segment, and information is new if it is not given. Discourse segments are informally defined as text paragraphs dealing with the same topic. As the texts used in the experiment are very small, they are assumed to consist of only one discourse segment. Words or phrases expressing information that is either new or contrastive are accented. Words or phrases expressing given information are normally unaccented, except when this information is contrastive. There is assumed to be no phonological difference between accents that express newness and those that express contrast. The exact landing place of the accent within an accented phrase is determined by Focus-Accent theory. The effect of the different combinations of newness, givenness and contrast is represented in Table 3.1, where a plus sign indicates that the corresponding phrase is accented, and a minus sign indicates that it is not.

	new	given
contrastive	+	+
¬ contrastive	+	-

TABLE 3.1: Information status and accentuation.

3.5.2 Method

The hypotheses presented in Section 3.5.1 were tested by means of a small perception experiment.⁹ In the experiment, twenty native speak-

⁹In order to test the hypotheses, both a perception and a production experiment might be used. However, production experiments have the practical disadvantage

ers of Dutch (14 male, 6 female, with different ages and backgrounds) were presented with twenty short texts in Dutch, displayed on a computer screen. Displayed next to each text were two buttons, which upon clicking played two different spoken versions of each text. The subjects were instructed to first read each text, and then listen to its two spoken versions. They could listen to each version as often as they liked. The spoken versions of the texts differed with respect to the accentuation of a target word in their final sentence: in one version (the ‘accented’ version), this word was accented, in the other version (the ‘unaccented’ version) it was not.

For each text, the subjects had to indicate which of its two spoken versions they found the most natural sounding. The subjects knew that the two versions only differed with respect to the pronunciation of the last sentence, but they were not told the exact nature of this difference, in order to reduce the awareness of the variable being tested.

3.5.3 Materials

The texts used in the experiment were constructed to test Hypotheses I and II, and consisted of two or (in most cases) three sentences. The first sentence of each text introduced a target item X. The last sentence of each text also contained a reference to X, which used the same wording. X was assumed to constitute given information by then. This was important for the experiment because, as Table 3.1 shows, words expressing given information are only accented if this information is contrastive. Thus, a subject’s preference for the accented version of a text could be interpreted as a preference for contrastive accent on the target word (= the final reference to X). If the target words expressed new information this interpretation would not follow, because new information is always accented. The texts were divided into three categories, representing three different types of context for the target word. An example from each category is given in Table 3.2.

The texts in Category I contained a reference to an alternative item Y, preceding the final reference to X, but there was no parallelism between the sentence introducing Y and the sentence containing the

that the results generally show a lot of variation and tend to be somewhat ‘vague’ (i.e., the subjects may use pitch accents of different strengths to highlight specific target items). As a consequence, the results of a production experiment tend to be much less clear-cut than those of a perception experiment. Therefore, a forced-choice perception experiment has been chosen here. It will be interesting, however, to conduct an additional production experiment and compare its results with those from the current perception experiment. This is a matter for future research.

I	<ol style="list-style-type: none"> 1. X A 2. Y B 3. X C 	<p><i>De burgemeester onthulde een standbeeld.</i> <i>Als dank kreeg de beeldhouwer een bloemetje.</i> <i>De burgemeester hield een toespraak.</i></p> <p>The mayor unveiled a statue. By way of thanks, the sculptor received a bouquet. The mayor made a speech.</p>
II	<ol style="list-style-type: none"> 1. X A 2. Y B P 3. X B Q 	<p><i>De burgemeester onthulde een standbeeld.</i> <i>Als dank kreeg de beeldhouwer een bloemetje.</i> <i>De burgemeester kreeg een fles wijn.</i></p> <p>The mayor unveiled a statue. By way of thanks, the sculptor received a bouquet. The mayor received a bottle of wine.</p>
III	<ol style="list-style-type: none"> 1. X A 2. B 3. X C 	<p><i>De burgemeester ging op vakantie naar Afrika.</i> <i>Het was daar erg warm.</i> <i>De burgemeester had veel last van de hitte.</i></p> <p>The mayor went on holiday to Africa. It was very hot there. The mayor suffered greatly from the heat.</p>

TABLE 3.2: Schematic representation and example text plus translation for each category. Here, {X, Y} and {P, Q} are pairs of alternatives; A, B and C are 'backgrounds'.

final reference to X.¹⁰ This corresponds to the context specified in Hypothesis I. Category I contained ten texts, two of which consisted of two sentences. The texts in Category II also contained a reference to an alternative item Y, and additionally they showed parallelism between the sentence introducing Y and the sentence containing the final reference to X. (In addition to {X, Y} these sentences contained another pair of alternatives {P, Q} and the backgrounds B of the two sentences were identical.) This corresponds to the context specified in Hypothesis II. Category II contained seven texts, one of which consisted of two sen-

¹⁰In one case, corresponding to example (27), the reference to Y preceded that to X within the same sentence.

tences. In nearly all texts, the alternative items were entities expressed by noun phrases. Only in one case, the items in the second pair of alternatives (i.e., the P and Q pair, not including the target item) corresponded to a property, expressed by adjectives. Finally, the texts in Category III contained no reference to an alternative item. This category was not directly related to the hypotheses, but was added to test the assumption that the presence of an intervening sentence does not affect givenness of an item. It contained only three texts, all of which consisted of three sentences.

Table 3.2 shows a schematic representation of the texts in each category, together with an example text and its translation. In the schematic representations, X represents the target item and Y its alternative; P and Q are another pair of alternatives; and A, B and C represent the remaining parts of the sentences (the ‘backgrounds’). In the example texts, the phrase *de burgemeester* (“the mayor”) refers to the target object X, and the phrase *de beeldhouwer* (“the sculptor”) refers to the alternative item Y. In the text of Category II, the phrases *een bloemetje* (“a bouquet”) and *een fles wijn* (“a bottle of wine”) form the other pair of alternatives {P, Q}. (Note that *een bloemetje* and *een fles wijn* are not alternatives because they share sortal properties, but because they both occur as arguments of *ontvangen* (“receive”).)

The spoken versions of the texts were generated using the speech synthesis system Calipso (Gigi and Vogten 1997, Klabbers 2000). Accents were assigned according to Focus-Accent theory. For the accented text versions, the target word was assumed to be in focus (leading to its accentuation), whereas for the unaccented versions this word was assumed to be defocused (and therefore was not accented). In some of the unaccented text versions, application of the Default Accent rule (see Sections 2.5.3 and 3.3) caused an accent to land on a word that would otherwise have remained unaccented (for instance, the verb *congratulated* in example (27)).

3.5.4 Results

The results of the experiment are shown in Table 3.3. For each category, the table indicates the average number of preferences for the unaccented and for the accented versions of the texts in that category, and whether the difference between the two is significant using the binomial test ($p < 0.05$). As Table 3 shows, there is a significant preference for the accented text versions in Category II. For the other two categories, the difference between preferences for the accented or the unaccented versions is not significant. For Category III, this is surprising since the target items in these texts were assumed to constitute given (and non-contrastive)

information, and therefore a preference for the unaccented text versions in this category was expected. An attempt to explain the unexpected results for Category III, as well as those for Category I, is made in Section 3.5.6.

	unaccented	accented	significance ($p < 0.05$)
I	10.9	9.1	-
II	4.7	15.3	+
III	11.0	9.0	-

TABLE 3.3: Average number of people that preferred either the accented or the unaccented versions of the texts in each category. Averages are computed over all texts within each category.

3.5.5 Discussion

The results for Categories I and II indicate two things. First, the mere presence of an alternative item does not trigger a significant preference for contrastive accent. Second, the presence of an alternative item combined with parallelism does trigger such a preference. In other words, Hypothesis I must be rejected, whereas Hypothesis II should be accepted. These findings might be explained as follows. Because the events that are described in the parallel sentences of Category II are similar, people feel the need to emphasise the differences between them by accenting the words that refer to the different participants in the events (i.e., the alternative items). On the other hand, the events described in the non-parallel sentences of Category I are not similar, and people feel no need to distinguish them by emphasising the alternative items. This is in line with the observations by Schmerling 1976, who points out that in example (28), the pronouns remain unaccented despite the reversal of semantic roles:

(28) John insulted Mary and then she SLAPPED him.

Although the above explanation accounts for the significant preference for accentuation in Category II, it does not explain the results for the other two categories. According to the ‘event-based’ account of contrast, the target items in Categories I and III are not contrastive, and therefore a significant preference for the unaccented text versions in these categories is expected. Nevertheless, Table 3.3 does not show a significant preference for either the accented or the unaccented text versions in Categories I and III.

It is important to note, however, that Table 3.3 shows the *average* preferences for accented and unaccented text versions, taken over all texts within Categories I and III. So, although the average preferences for accented and unaccented versions are nearly the same, this need not be the case for the preferences associated with the individual texts in these categories. In fact, there are large differences in preference within Categories I and III: for some texts in these categories, there is a strong (but not always significant) preference for the unaccented version, and for other texts there is a preference for the accented version. Since these differences within Categories I and III cannot be explained by the presence or absence of an alternative item or parallelism, another factor must play a role here. In Section 3.5.6 it is argued that this factor may be *coherence*.

3.5.6 Coherence

As argued in Section 3.5.1, the texts used as materials in the experiment are too short to be divided into more than one discourse segment. This means that the accentuation of supposedly given items in Categories I and III cannot be explained through the presence of discourse segment boundaries that limit the givenness of discourse items. Still, it may be the case that *local* (i.e., intra-segmental) discourse structure plays a role here, with the local relations between sentences influencing the degree of givenness of discourse items. The argument presented in this section is that sharp transitions between sentences create an effect of local incoherence, causing previously mentioned items to be ‘displaced’ and therefore accented (see Section 3.2 for a discussion of the effect of displacement).

A well-known theory of local discourse structure is Centering Theory (Grosz et al. 1995). Centering Theory assigns to each utterance U_i a set of *forward looking centers* $C_f(U_i)$ containing representations of the entities referred to in U_i . This set is partially ordered to reflect the relative prominence of the referring expressions within the utterance. Additionally, each utterance U_i is (in principle) assigned a *backward looking center* $C_b(U_i)$. This is an entity which was also mentioned in the preceding utterance U_{i-1} , and is therefore a member of $C_f(U_{i-1})$. The C_b functions as a kind of link between two utterances, establishing coherence between them. For example, in the sequence *The dog chased the cat. The cat chased a mouse*, the C_b of the second sentence is the cat. Centering Theory provides a typology of transitions between utterances to model the coherence of a discourse segment. The type of transition between utterances U_i and U_{i-1} depends on the relation of $C_b(U_i)$ to

	accented	unaccented
IC	12,2	7,8
C	7,2	12,9

TABLE 3.4: Average number of people that preferred either the accented or the unaccented versions of the texts in each coherence class, combined for Categories I and III. Averages are computed over all texts within each class (5 IC texts and 8 C texts).

both $C_b(U_{i-1})$ and $C_f(U_i)$. The basic types of transition, ordered by decreasing coherence, are CONTINUE, RETAIN, SMOOTH SHIFT and ROUGH SHIFT. If none of the entities mentioned in utterance U_i have been mentioned in U_{i-1} (i.e., $C_f(U_i) \cap C_f(U_{i-1}) = \emptyset$), then U_i has no C_b . In this case, the coherence between U_i and U_{i-1} is obviously minimal.¹¹

A Centering analysis of the texts used in the experiment reveals that these texts can be divided into two basic coherence classes, depending on the presence or absence of a C_b in their final sentence (note that this way, only the coherence between the *last two* sentences of the discourse segment is taken into account). Texts that do not have a C_b in their final sentence may be classified as ‘incoherent’ (IC), and texts that do have a C_b in their final sentence may be classified as ‘coherent’ (C).¹² In each of the Categories I to III, approximately half of the texts belongs to the C class, and half of the texts belongs to the IC class. In Category II, with the parallel texts, there appears to be no relationship between accentuation and coherence class: in this category, the accented version was preferred for all texts except one (an IC text, where the two spoken versions scored equal). This suggests that the effect of parallelism is strong enough to overrule any effects of coherence. For this reason, in the remainder of this section only the texts of the combined Categories I and III are considered.

For the texts in Categories I and III, there does seem to be a relation between coherence class and accentuation preference. The average accentuation preferences per coherence class are shown in Table 3.4. Al-

¹¹The presence of a C_b is assumed to be optional, as in Kameyama 1986. If the presence of a C_b is considered to be obligatory, then the cases where $C_f(U_i) \cap C_f(U_{i-1}) = \emptyset$ would be interpreted as instances of a ROUGH SHIFT, the least coherent of Centering relations. This would not influence the main line of argumentation followed in the remainder of this section.

¹²According to Centering Theory, some texts in the C class are more coherent than others, depending on the type of transition involved (CONTINUE, RETAIN or SHIFT). The differences in coherence within the C class are relatively minor and therefore are ignored here.

though the differences in preference are non-significant, a clear trend is visible: coherence of a text (i.e., the presence of a C_b in the final sentence) leads to a preference for its unaccented version, and incoherence (i.e., the absence of a C_b) leads to a preference for the accented version.

The perceived relationship between local coherence and accentuation can be explained as follows. Since an utterance which lacks a C_b is not explicitly linked to its predecessor, the transition between them is rather abrupt, creating a kind of local ‘barrier’. The effect of such an intra-segmental barrier appears to be similar to that of a segment boundary, in that the discourse entities that were mentioned before the boundary become less accessible, and no longer count as fully given. When such entities are referred to again, they are accented to mark their reintroduction. This is similar to the effect of ‘displacement’ found by van Donzel 1999 (see Section 3.2). In fact, the target items in the IC texts are all displaced in the sense of van Donzel (i.e., an item is displaced if it is impossible to express it using a pronoun). According to Centering Theory, if an utterance U_i contains any pronouns, then the $C_b(U_i)$ must also be expressed by a pronoun. From this it follows that in the absence of a $C_b(U_i)$, no pronouns can occur in U_i .

There are insufficient data available here to make a substantial claim about the link between local incoherence and accentuation of given items, but as the current (sparse) data are in line with the findings of van Donzel concerning displaced discourse items, it seems safe to assume that the slight preference for the accented versions of the IC texts in Categories I and III may be attributed to decreased accessibility (or givenness) of the target object, not contrast.

3.5.7 Conclusions

Summarising, the experiment has shown that parallelism causes a significant preference for contrastive accent, and that the mere presence of an alternative item does not. Presumably, this is because people prefer to emphasise the points of difference between similar events by assigning contrastive accents to the different actors in these events. In addition, it has been observed that people show a slight preference for accenting previously mentioned items that are ‘displaced’; these items appear to be regarded as ‘new’ again. In texts that are coherent and show no parallelism, people tend to prefer deaccentuation of previously mentioned items. In texts that are coherent, but do show parallelism, there is no such tendency: here, there is a strong preference for accentuation of given items. Unlike the effect of parallelism, the observed effects of (in)coherence are not significant; more experimental data are needed to

come to conclusive results concerning the relation between accentuation and local coherence.

3.6 Assigning contrastive accent within the LGM

The current section presents a practical method for the generation of contrastive accent within the LGM, which has been successfully implemented in the GoalGetter system. This method is not based on any of the formal approaches to contrast discussed in Section 3.4, as these could not be straightforwardly implemented in the LGM. Instead of looking at the semantic representations of generated sentences, the chosen method tries to establish parallelism between sentences on the level of the data structures which serve as the basis for generation within the LGM. The experiment described in the previous section led to the conclusion that people prefer to have a contrastive accent on different arguments of the same verb. This idea is taken one step further here: contrastive accent is assumed to be triggered by the subsequent mention of similar events, which may or may not be expressed using the same verb.

3.6.1 Deriving contrast from data structures

As argued in Section 3.2, the original LGM must be extended with a way of assigning contrastive accents. In Section 3.4, several formal approaches to contrast have been discussed, which are possible candidates for implementation. A common feature of these approaches is that in some way or other, they make use of semantic sentence representations. This poses a problem for their implementation in the LGM, because the syntactic templates used in the LGM are not associated with semantic representations. The only sentence representations that are available within LGM are the syntactic structures that are present in the templates. These are not quite suitable for determining contrast, however, since – as originally argued by Ladd 1980 in opposition to the claim by Chomsky 1971 (see Section 3.3) – contrast appears to be a matter of semantic rather than syntactic parallelism. In fact, not even parallelism of semantic representations is required in order for two sentences to be contrastive, as is shown by examples like (20), (24) and (29) below, which is cited by Delin and Zacharski 1997.

(29) When I arrived back in Croton I bumped into Laesus, though I had not expected to see him again and HE looked pretty surprised at seeing ME.

(Accents marked in original: Lindsey Davis 1991. *Shadow in bronze*. London: Pan Books. p. 98)

In this example, neither the syntactic nor the semantic representa-

tions of the two sentences are parallel. Still, as argued by Delin and Zacharski 1997, the sentences clearly stand in contrast to each other at the level of the *events* that are being expressed. In applications based on the LGM, like GoalGetter, such ‘event-based’ contrasts occur very frequently. The LGM-based applications typically cover a limited domain, involving only a small number of different events, for example “scoring a goal” or “receiving a card” in the GoalGetter system. The language generation technique embodied in the LGM is aimed at expressing this limited number of events in many linguistically varied ways. In the first place, a specific event can be described using any of a number of different syntactic templates. Additionally, the gaps in each template can be filled in different ways, resulting in different sentences. As a consequence, two sentences that are neither syntactically nor semantically parallel may express the same kind of event, as illustrated by example (30) from GoalGetter.

- (30) *De score werd in de zesde minuut geopend door Koeman van Feyenoord. ACHT minuten LATER maakte KLUIVERT een doelpunt voor AJAX.*

Translation:

The score was opened in the sixth minute by Koeman from Feyenoord. EIGHT minutes LATER, KLUIVERT scored a goal for AJAX.

In this example, both sentences describe the scoring of a goal. There is a clear contrast between the two events: in the first case, the goal was scored by Koeman for Feyenoord, whereas in the second, it was scored (eight minutes later) by Kluyvert for Ajax. Contrastive sequences such as (30) are very common in the texts generated by GoalGetter; in Section 3.6.2, some others are discussed.

All these examples indicate that what is relevant for contrast is not so much the form and wording of an utterance, but the kind of event that is being expressed. In addition, examples like (14) indicate that it is not the presence of another item of the same type that causes an item to be contrastive, but that *any* two items that have the same thematic roles (e.g., agent or patient) with respect to the same verb may be considered as contrastive. Another example is (26) from the experiment described in section 3.5. The subjects almost unanimously chose to accent the pronoun *him* (referring to the pope) even though in isolation “the pope” and “the ground” would never be regarded as alternatives of each other. In the experiment, contrastive accent was consistently assigned to items that were involved in similar events.

So, detection of contrast in LGM-based applications should not depend on the surface form of the generated utterances, nor on the presence of items that are ‘of the same type’, but on the kinds of events that are being expressed. These events may be regarded as the ‘deep’ semantics, or ‘conceptual representation’ of the sentence.¹³ Most approaches to contrast detection discussed in this chapter require parallelism at the level of semantic representations. In the current section, however, it is argued that contrast between sentences arises from a parallelism at the conceptual level. In the LGM, the data structures expressed by the generated sentences may be seen as the conceptual representations of these sentences, and contrast can be detected directly by looking at these data structures. This includes cases of ‘indirect’ contrast, where the contrastive sentences lack parallelism at the syntactic and semantic level. The method of contrast detection in the LGM which is proposed here, is based on the simple principle that two consecutive sentences which express the same type of data structure (and therefore express similar information) are potentially contrastive:

Definition 1 (Contrast)

1. Two consecutive sentences are contrastive if they express contrastive data structures
2. Two data structures are contrastive if
 - a. they are of the same type *T* and
 - b. they have contrastive values for at least one attribute *A* of *T*
3. Two values are contrastive if
 - a. they are non-identical primitives *or*
 - b. they are contrastive data structures

Data structures consist of a number of attributes, which may either have a ‘primitive’ value (i.e., a value that has no internal structure) or a complex value (i.e., a value which is itself a data structure). In the latter case, part 2 of Definition 1 is applied recursively.

¹³This view is related to the assumption of a *two-level semantics* by Bierwisch 1983, who distinguishes a semantic and a conceptual level of representation. According to this approach, semantic representations are closely linked to grammar, and thus to the surface form of a sentence, whereas conceptual representations are language-independent mental representations of world knowledge. (See also Bateman 1992 for arguments in favour of this ‘stratified’ approach to semantics.)

Given Definition 1, the proposed method for assigning contrastive accent can be described as follows:

Assignment of contrastive accent: if two consecutive sentences S_{i-1} and S_i are contrastive as defined in Definition 1, the constituents of S_i that express contrastive values should be marked [+C], indicating that they are *in focus* due to contrast.

As discussed in Chapter 2, constituents that are marked [+C] must receive a pitch accent, the landing place of which is decided on the basis of (a version of) Focus-Accent theory. They cannot be deaccented due to givenness.

It should be noted that only the relevant constituents of S_i , the *second* sentence in a sequence of two, are marked [+C]. The lack of pre-planning in the LGM, discussed in Section 2.4.3, makes it impossible to assign contrastive accents to the first sentence (S_{i-1}) of a pair of contrastive sentences. As explained in Chapter 2, the LGM can look back upon the monologue history, but there is no text planning beyond the sentence currently being generated. This means that during the generation of a sentence, it is not yet known whether the following sentence will stand in contrast to it; the only thing that can be determined is whether the current sentence is contrastive to its predecessor. As a consequence, contrastive accents can only be assigned to the second sentence in a contrastive pair, and not to the first sentence. This is comparable to the situation where a speaker has not yet thought of what to say next when he utters the first of two contrastive utterances. Thus, restricting the placement of contrastive accents to the second sentence in a contrastive pair appears to be not unrealistic. Still, it might be a useful extension to the LGM to postpone the assignment of prosodic markers to a sentence (and consequently also its pronunciation) until it has been determined what the next sentence will be; this will allow for bidirectionality of contrastive accent assignment (and maybe also for deaccenting, as suggested in van Deemter 1998; see also Section 3.6.3).

In Section 3.6.2, the mechanism for contrast determination based on data structures is illustrated using some examples from the GoalGetter system.

3.6.2 Contrastive accent assignment: illustrations

The translation of example (4) from Section 3.2 will be used as an illustration of the approach to contrast outlined above. It is repeated below in (31).

- (31) After three minutes, Feyenoord took the lead through a goal by Koeman. In the sixteenth minute, Kluivert kicked a penalty home for Ajax. Ten minutes later, Larsson scored for Feyenoord.

As was explained in Section 2.4.1, GoalGetter’s football reports are generated on the basis of a typed data structure derived from the information on a Teletext page. The field goallist of this data structure contains a sequence of structures of type *goal_event*, each specifying the team for which a goal was scored, the player who scored, the time and the kind of goal: normal, own goal or a goal resulting from a penalty. The last two sentences of example (31) both express such a *goal_event*. These sentences and their data structures are shown in Figure 3.2. For simplicity, the values of the attributes team and player are represented as primitives, corresponding to the names of the team and the player. In reality, these values are complex data structures consisting of a number of attributes (see example (34) below).

As can be seen in Figure 3.2, each attribute of the *goal_event* expressed by the final sentence of (31) has a value that is different from the corresponding value in the preceding *goal_event*. This means that the two data structures are contrastive, as defined in Definition 1: they are of the same type (*goal_event*), and they have contrastive (i.e., non-identical) values for at least one – in fact, all – of their attributes. Therefore, the two sentences are regarded as contrastive, even though they do not use the same verbs. Since all attributes of the two data structures

$$\left[\begin{array}{l} \textit{goal_event} \\ \text{team :} \quad \text{Ajax} \\ \text{player :} \quad \text{Kluivert} \\ \text{minute :} \quad \text{16} \\ \text{goaltype :} \quad \text{penalty} \end{array} \right]$$

In the sixteenth minute, Kluivert kicked a penalty home for Ajax.

$$\left[\begin{array}{l} \textit{goal_event} \\ \text{team :} \quad \text{Feyenoord} \\ \text{player :} \quad \text{Larsson} \\ \text{minute :} \quad \text{26} \\ \text{goaltype :} \quad \text{normal} \end{array} \right]$$

Ten minutes later, Larsson scored for Feyenoord.

FIGURE 3.2: Data structures expressed by the two last sentences of (31).

have contrastive values, all phrases in the final sentence expressing these values should receive contrastive accent. These are the time expression *Ten minutes later* (expressing the minute attribute), the proper name *Larsson* (expressing the player) and the proper name *Feyenoord* (team), which must be accented despite being given. The value of the goal-type attribute is not expressed in the surface structure of the sentence; however, if it were, it would receive contrastive accent (e.g., *Ten minutes later, Larsson scored a NORMAL goal for Feyenoord*). Similarly, if the team name *Ajax* had not been mentioned in the second sentence (for instance, because it was considered common knowledge that Kluivert played for Ajax), *Feyenoord* would still have received an accent, resulting in the following accent assignment:

- (32) After three minutes, Feyenoord took the lead through a goal by Koeman. In the sixteenth minute, Kluivert kicked a penalty home.
TEN minutes LATER, LARSSON scored for FEYENOORD.

In the above example, not accenting the second *Feyenoord* – as the LGM would do without the contrast algorithm – would probably be very confusing to the listener, since the resulting accentuation pattern would only be appropriate if all goals were scored for Feyenoord. A listener who did not know Kluivert might get the impression that Kluivert played for Feyenoord.

Another, somewhat more complicated illustration of this form of implicit contrast is the sequence in (33), a translation of an actual text fragment generated by GoalGetter. It is shown with (a) the accentuation pattern as it used to be generated by the LGM's original accentuation algorithm and (b) the accentuation pattern generated by the new accentuation algorithm, which includes determination of contrast as sketched above.

- (33) a. In the sixteenth minute, the Ajax player Kluivert kicked the ball into the wrong goal. TWENTY minutes LATER, OVERMARS scored for Ajax.
b. In the sixteenth minute, the Ajax player Kluivert kicked the ball into the wrong goal. TWENTY minutes LATER, OVERMARS scored for AJAX.

As shown in (33)a, the original accentuation algorithm would deaccent the second reference to *Ajax* due to givenness. This results in a confusing accentuation pattern: on the one hand the listener knows that scoring an own goal means scoring for the opposing team (i.e., *not Ajax*), but on the other hand the accentuation pattern suggests that both Kluivert and Overmars had scored for Ajax. In contrast, the ac-

$$\left[\begin{array}{l} \textit{goal_event} \\ \text{team :} \quad \text{Feyenoord} \\ \text{player :} \quad \text{Kluivert} \\ \text{minute :} \quad \text{16} \\ \text{goaltype :} \quad \text{own} \end{array} \right]$$

In the sixteenth minute, the Ajax player Kluivert kicked the ball into the wrong goal.

$$\left[\begin{array}{l} \textit{goal_event} \\ \text{team :} \quad \text{Ajax} \\ \text{player :} \quad \text{Overmars} \\ \text{minute :} \quad \text{36} \\ \text{goaltype :} \quad \text{normal} \end{array} \right]$$

Twenty minutes later, Overmars scored for Ajax.

FIGURE 3.3: Data structures expressed by (33).

centuation pattern in (33)b does not produce such a clash. It actually seems to make it easier for the hearer to process this text fragment. The accent on the second *Ajax* emphasises that the previous goal was not for Ajax, and thus helps to guide the hearer to the correct interpretation. As in the previous example, the required contrastive accents can be immediately derived from the data structures expressed by the subsequent sentences in (33), given in Figure 3.3. A simple inspection of the two team attributes reveals that they have different values, and so the phrase expressing the team attribute in the final sentence of (33) should receive contrastive accent, even though the corresponding value of the previous sentence was not overtly expressed.

The definition of contrastive data structures given in section 3.6.1 allows ‘contrastive values’ to be determined recursively: a contrastive value may correspond either to a primitive or to a data structure, which in its turn contains at least one contrastive value. In this way, contrastive values occur at different levels in the data structure. The effects of this mechanism can be illustrated using the following example, of which the corresponding data structures are given in Figure 3.4.

- (34) In the twenty-third minute, the Nigerian player Kanu scored a goal for Ajax. Eleven minutes later Larsson, the Swedish forward, scored for Feyenoord.

Figure 3.4 shows the actual values of the player attributes, which are data structures instead of primitives. (In Figures 3.2 and 3.3, simplified

<i>goal_event</i>									
team:	Ajax								
player	<table style="border-collapse: collapse; margin-left: 20px;"> <tr> <td style="border-left: 1px solid black; border-right: 1px solid black; padding: 2px;">first_name :</td> <td style="padding: 2px;">Nwankwo</td> </tr> <tr> <td style="border-left: 1px solid black; border-right: 1px solid black; padding: 2px;">last_name :</td> <td style="padding: 2px;">Kanu</td> </tr> <tr> <td style="border-left: 1px solid black; border-right: 1px solid black; padding: 2px;">nationality :</td> <td style="padding: 2px;">Nigerian</td> </tr> <tr> <td style="border-left: 1px solid black; border-right: 1px solid black; padding: 2px;">position :</td> <td style="padding: 2px;">forward</td> </tr> </table>	first_name :	Nwankwo	last_name :	Kanu	nationality :	Nigerian	position :	forward
first_name :	Nwankwo								
last_name :	Kanu								
nationality :	Nigerian								
position :	forward								
minute:	23								
goaltype:	normal								

In the twenty-third minute, the Nigerian player Kanu scored a goal for Ajax.

<i>goal_event</i>									
team:	Feyenoord								
player	<table style="border-collapse: collapse; margin-left: 20px;"> <tr> <td style="border-left: 1px solid black; border-right: 1px solid black; padding: 2px;">first_name :</td> <td style="padding: 2px;">Henryk</td> </tr> <tr> <td style="border-left: 1px solid black; border-right: 1px solid black; padding: 2px;">last_name :</td> <td style="padding: 2px;">Larsson</td> </tr> <tr> <td style="border-left: 1px solid black; border-right: 1px solid black; padding: 2px;">nationality :</td> <td style="padding: 2px;">Swedish</td> </tr> <tr> <td style="border-left: 1px solid black; border-right: 1px solid black; padding: 2px;">position :</td> <td style="padding: 2px;">forward</td> </tr> </table>	first_name :	Henryk	last_name :	Larsson	nationality :	Swedish	position :	forward
first_name :	Henryk								
last_name :	Larsson								
nationality :	Swedish								
position :	forward								
minute:	33								
goaltype:	normal								

Eleven minutes later Larsson, the Swedish forward, scored for Feyenoord.

FIGURE 3.4: Data structures expressed by (34).

values were shown for the player attributes.) Some of the values of the attributes in these player data structures are used in the referring expressions for the players, e.g., the description *the Nigerian player Kanu* includes the values for the *last_name* and *nationality* attributes. Now, using Definition 1, it can be determined that not only the values for the *goal_events*' player attributes in Figure 3.4 are contrastive, but also the values for the players' attributes *first_name*, *last_name* and *nationality*. This means that in the second sentence, not only the expression for the player as a whole (*Larsson, the Swedish forward*) should be marked as contrastive, but also the expressions for the *last_name* and *nationality* values (*Larsson* and *Swedish*, respectively).¹⁴ This is shown in Figure 3.5.

¹⁴In line with Ladd 1980, van Deemter (p.c.) suggests that in cases with multiple

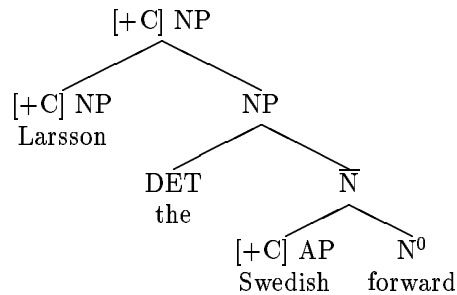


FIGURE 3.5: Syntactic tree for the NP *Larsson, the Swedish forward*.

The word *forward* does not express a contrastive value, and is therefore not marked [+C]. Still, this word does receive an accent due to ‘newness’ (since it has not been previously mentioned), which results in the accentuation pattern LARSSON, *the* SWEDISH FORWARD.

The approach described in this section is in keeping with the results of the experiment discussed in Section 3.5; contrastive accent is triggered by the subsequent mention of similar events. The mere presence of an alternative item in the preceding context does not trigger the assignment of contrastive accent. For example, in example (35) below, the second and third sentences will not be regarded as contrastive, since the third expresses a *goal_event* but the second does not. Therefore, despite the presence of the alternative *Ajax*, the contrast algorithm will not assign an accent to *Feyenoord* in the final sentence, as shown in (35).

- (35) After three minutes, FEYENOORD took the lead through a goal by KOEMAN. This caused AJAX to fall behind. Ten minutes later LARSSON scored for Feyenoord.

A similar approach to the assignment of contrastive accent is advocated by Delin and Zacharski 1997, who discuss the generation of intonation contours in the BRIDGE spoken dialogue system. Inspired by Schmerling 1976, they say that “for a discourse entity that represents an eventuality, the speaker determines what distinguishes this eventuality from others” (and accents the associated expressions). How this strategy is modelled in their system, is not explained in the paper.

contrasts, as in the current example, the speaker may be free to choose which contrasts to realise. In particular, when there is a high number of (layered) contrasts, it seems likely that not all words expressing these need to be accented. However, modelling such a choice between contrasts falls outside the scope of the research presented here.

3.6.3 A possible extension

An interesting further move would be to extend the current approach so as to deal with deaccenting as well.¹⁵ The parts of a sentence that express values which are identical to values in the data structure of the previous sentence, should then be deaccented. This would make it possible to account for cases of deaccenting that cannot be handled by the LGM's current defocusing strategy, described in Section 2.5.2. This can be illustrated by example (36). The corresponding data structures are given in Figure 3.6. These structures are of type *card_event*, and describe at which time which player received a card of which colour.

- (36) In the fifth minute, Koeman was sent off the field. KLUIVERT received a red card in the TWELFTH minute.

$$\left[\begin{array}{l} \textit{card_event} \\ \text{player :} \quad \text{Koeman} \\ \text{minute :} \quad 5 \\ \text{cardtype :} \quad \text{red} \end{array} \right]$$

In the fifth minute, Koeman was sent off the field.

$$\left[\begin{array}{l} \textit{card_event} \\ \text{player :} \quad \text{Kluivert} \\ \text{minute :} \quad 12 \\ \text{cardtype :} \quad \text{red} \end{array} \right]$$

Kluivert received a red card in the twelfth minute.

FIGURE 3.6: Data structures expressed by (36).

The first sentence of (36) expresses its data structure in an indirect manner, leaving the colour of the card unspecified but inferable. The final sentence of (36) does explicitly mention the colour. Because the events described in the two sentences both involve the same type of card, the phrase expressing this type (*red card*) in the final sentence should be deaccented due to *concept-givenness* (see Section 2.5.2). The concept-givenness of *red card* is not detected by the LGM's defocusing algorithm, since in the first sentence the type of card is not explicitly mentioned. However, by looking at the data structures in Figure 3.6, we can see that the values of the card feature are identical, and that the phrase *red card*

¹⁵The link between parallelism and deaccenting has been pointed out by Gardent 1997;2000. The extension described in this section has not been implemented.

in (36) should be deaccented. This results in the correct accentuation pattern as shown in (36), which will confirm the inference of the hearer that Koeman was shown a red card too.

Applying the deaccentuation strategy described above to example (34), discussed in Section 3.6.2, will cause the word *forward* to be deaccented due to givenness, as both Kanu and Larsson are forwards. This results in the following accentuation pattern for the second sentence:

(37) In the twenty-third minute, the Nigerian player Kanu scored a goal for Ajax. ELEVEN minutes LATER LARSSON, the SWEDISH forward, scored for FEYENOORD.

Deaccentuation of *forward* in (37) is quite acceptable if we assume the hearer to be aware of the fact that the player Kanu is a forward, just like Larsson. If the hearer does not have this previous knowledge, deaccenting the word *forward* might lead the hearer to ‘accommodate’ (Lewis 1979) the fact that Kanu is also forward, which is the intended effect. Alternatively, deaccenting *forward* might simply confuse the hearer. It remains to be investigated which of the two effects is most likely to occur.

So, while the approach to deaccentuation discussed in this section looks promising in some respects, in particular the treatment of ‘implicit’ givenness, its full impact cannot yet be overseen. This is a matter for future research.

3.6.4 Discussion

An important advantage of using data structures rather than syntactic or semantic representations to assign contrastive accents within the LGM is that it allows for the detection of contrast in cases where there appears to be no direct syntactic or semantic parallelism between sentences. Such cases occur quite frequently in LGM-based applications, and the approaches discussed in Section 3.4 are not well suited to deal with these (at least not without the use of additional knowledge sources such as meaning postulates). The proposed method for contrast detection in the LGM requires no explicit listing of semantic equivalences or entailments, since the relevant information for the detection of contrast is encoded in the data structures from which the sentences are generated; whether this information is then expressed explicitly in the actual sentences is irrelevant. This means that contrasts between consecutive sentences that do show syntactic or semantic parallelism can also be accounted for, if these sentences express similar events. Those contrasts are then simply sub-cases of the more general case.

On the other hand, the approach proposed here is limited in that it

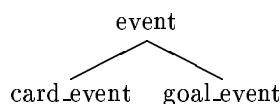


FIGURE 3.7: Event hierarchy in GoalGetter.

cannot account for cases of contrast where two sentences are syntactically parallel, but do not express the same kinds of events. This may occur for instance when the same verb is used in different senses. In those cases, the contrast is not situated at the level of data structures but at the level of surface structure. Consequently, Definition 1 cannot capture such contrasts. Instead, they may be detected by comparing the syntactic representations of generated sentences, which are available from the templates in the LGM. Contrastive constructions that are purely syntactic in nature appear to be quite rare. Therefore, no method for detecting them has been implemented in the LGM, although it would not be difficult to do so given the availability of full syntactic sentence structures in the LGM.

In addition, it is clear that the method for contrast determination described in this section places a great responsibility on the data structures that are used. The problem of defining parallelism is arguably shifted to the design of the data structures: these must be set up in such a way that potentially contrastive items get assigned identical data types. It is not always clear, however, which items should be regarded as ‘potentially contrastive’. For example, in the GoalGetter application a distinction is made between `card_events` and `goal_events`, as depicted in Figure 3.7. Following Definition 1, `goal_events` are potentially contrastive to other `goal_events` but not to `card_events`. However, other classifications of events could be made as well, e.g., by adding separate data types for normal goals, own goals and penalties. In that case, the type of a goal would be directly reflected in its data structure type, instead of being specified as the value of the `goaltype` attribute of a `goal_event`.¹⁶ Since in this situation, depicted in Figure 3.8, a penalty and a normal goal would be represented by different types of data structures, two sentences expressing those events would not count as contrastive. As a consequence, no contrastive accent would be predicted in cases like our earlier example (31), for which the alternative data structures are given in Figure 3.9. Clearly, this is not a desirable result. Of course, Definition 1 could be relaxed so that two events having the same parent (or even grand-

¹⁶Similarly, `card_events` could be divided into `red_card_events` and `yellow_card_events`. This is not shown.

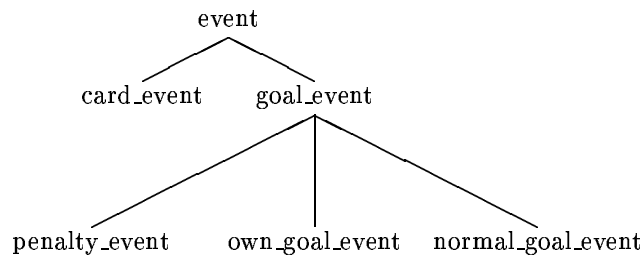


FIGURE 3.8: Alternative event hierarchy.

<i>penalty_event</i> team : Ajax player : Kluivert minute : 16

In the sixteenth minute, Kluivert kicked a penalty home for Ajax.

<i>normal_goal_event</i> team : Feyenoord player : Larsson minute : 26

Ten minutes later, Larsson scored for Feyenoord.

FIGURE 3.9: Alternative data structures for the last two sentences of (31).

parent) type would count as potentially contrastive, but this seems to be too permissive as it would allow any two events in GoalGetter to be regarded as potentially contrastive.

We see that the success of the method for contrastive accent assignment presented here depends largely on the level of specification of the data structures; for instance, the classification given in Figure 3.8 appears to be too specific for contrast prediction in the football domain, whereas the more global classification in Figure 3.7 gives better results. This suggests that there is a level of specificity that is most suitable for use in the determination of contrast. It seems likely that this level corresponds to the ‘basic conceptual level’, the preferred level of categorisation discussed by Cruse 1977, Rosch 1978, Levelt 1989 and others.¹⁷

¹⁷See Chapter 4 for the use of basic levels in the generation of referring expressions.

The notion ‘basic level’ is mostly used in the classification of objects (e.g., people prefer the basic level description ‘apple’ over the more general ‘fruit’ or the more specific ‘Jonagold’), but it can also be used in the classification of events, where for instance ‘walking’ can be regarded as the basic level between ‘moving’ (too general) and ‘sauntering’, ‘striding’, and so on (too specific). Similarly, it could be argued that ‘goal scoring’ is a basic level concept whereas, e.g., ‘penalty scoring’ is not.

In short, the choice of data structures is essential for the result of contrast detection in the LGM, and therefore the data structures should be designed in a way that is psychologically realistic. A plausible requirement is that data structures should be typed at the level of ‘basic level categories’. As a resource for determining these categories, the online lexical database WordNet might be used (Miller 1995, Fellbaum 1998; see also <http://mind.princeton.edu/wordnet/>). WordNet is based on theories of human lexical memory and has a large coverage of English nouns, verbs, adjectives and adverbs. For nouns and verbs, basic concept levels are specified.¹⁸

Of course, the importance of correct classification is not unique for the data structure-based approach that is advocated here; it also holds for the approaches to contrast described in Section 3.4 (except the contrariety-approach of van Deemter): in order to determine which items count as alternatives, a suitable ontology is indispensable. In fact, almost all systems that deal with natural language processing or generation have to make use of some kind of ontology to represent world knowledge. For most practical applications, a relatively small, domain-specific ontology may be sufficient, but – as argued above – care should be taken to set this up in a psychologically realistic way. At the same time, since application domains are continuously getting larger, there is an increasing interest in the development of extensive, general ontologies that are re-usable in different systems. An overview of ontologies of various kinds is presented by Bateman 1992, who discusses the different problems and requirements related to their organisation. It would be interesting to investigate the suitability of different existing ontologies for contrast detection; however, this must be left for future research.

3.7 Summary

In this chapter, it has been shown that the strategy of de-accenting given information can lead to the generation of incorrect accentuation

¹⁸The WordNet database could also be used to retrieve other semantic information that is relevant for the computation of prosody, as it is organised around semantic relations that include synonymy, hypernymy, and antonymy. See Hiyakumoto et al. 1997 for a discussion of this use of WordNet.

patterns if contrast is not taken into account. Contrastive information should receive an accent, even if it is given. Therefore, a method to detect contrast is required for prosody computation.

Most early works on contrast offer informal definitions in terms of parallelism or pairs of alternatives. More recent, formal approaches to the detection of contrast, proposed by Prevost 1995, Pulman 1997, and van Deemter 1994a;1998;1999 make use of these notions as well. The relation between parallelism and contrast was investigated by means of a small perception experiment. In the experiment, two sentences were counted as parallel if they were identical except for their 'alternative items', i.e., the different arguments of the verb. The experimental results indicate that parallelism indeed causes a significant preference for contrastive accent, and that the mere presence of an alternative item does not. Apparently, people prefer to emphasise the points of difference between similar events by assigning contrastive accents to the different actors in these events.

This idea forms the basis for a practical method to detect contrast in the LGM, which does not make use of surface structures or semantic representations. Instead, assignment of contrastive accent is based on the data structures expressed by the generated sentences: if two consecutive sentences express the same type of data structure, they are potentially contrastive, and contrastive accent is assigned to the phrases that express the different parts of the data structures. It should be noted that this method places a high responsibility on the data structures used as the basis for language generation, and that further research into the requirements on these data structures is still needed. This, however, falls outside the scope of this thesis. The proposed method for contrast detection has been implemented in the GoalGetter system.

Generating descriptions in context

4.1 Introduction

A language generation system uses natural language to express non-linguistic data, consisting of information about entities and the relations between them. One of the essential tasks in language generation is the generation of phrases referring to those entities. The easiest way of doing this is by using an entity's *name*, if it has one. However, always using proper names to refer to entities is likely to result in output texts that are boring and unnatural. In certain linguistic contexts other types of expression are preferable, e.g., pronouns. In addition, not all entities have names. This means that a natural language generator should be able to generate other types of expressions as well, e.g., definite descriptions and pronouns. The main focus of this chapter is on the generation of definite descriptions of the form “the \bar{N} ”, where \bar{N} expresses a number of properties of the intended referent (e.g., *the Swedish forward*). Other types of expressions, such as pronouns and indefinite descriptions, are only briefly discussed.

As pointed out in Section 2.4.5, the LGM uses a fairly *ad hoc* strategy for the production of definite descriptions. Only anaphoric use of definite descriptions is allowed, which means that the LGM cannot use a definite description to introduce an entity to the discourse. In addition to this limitation, the LGM does not have a principled method for determining which properties to include in a description. These limitations do not cause any real problems in the LGM-based systems that have so far been built, because most entities in those systems have proper names that can be used for identification. For possible future applications, a more sophisticated algorithm for the generation of definite descriptions is indispensable, however.

A well-known algorithm for the generation of definite descriptions is the Incremental Algorithm, described in Reiter and Dale 1992 and

Dale and Reiter 1995. They focus on the generation of *distinguishing* descriptions, i.e., definite descriptions that uniquely identify one of the entities in the application domain, providing “an accurate characterisation of the entity being referred to, but not of any other entity in the current context set” (Reiter and Dale 1992:232). In their 1995 paper, Dale and Reiter present various algorithms for the generation of distinguishing descriptions, which they developed alone or in tandem. They argue that their *Incremental Algorithm*, discussed in more detail below, is the best one from a computational point of view (it has a low complexity and is fast) as well as from a psychological point of view (humans appear to do it in a similar way).

Although Dale and Reiter 1995 primarily aimed at investigating the computational implications of obeying the Gricean maxims¹, the Incremental Algorithm has become more or less accepted as the state of the art for generating definite descriptions. It is therefore an obvious candidate for implementation in the LGM. However, due to Dale and Reiter’s original motivation, various other aspects of the generation of definites remained somewhat underdeveloped. In this chapter, a number of these aspects are fleshed out, without losing sight of the attractive properties of the original algorithm (speed, complexity, psychological plausibility).

Closely related in spirit to the current work are Horacek 1997 and Stone and Webber 1998. Horacek 1997 makes various interesting observations about the limitations of the Incremental Algorithm and its ilk and proposes a new algorithm with ‘flexible interfaces’ to other modules. Many of the issues discussed by Horacek are also addressed in this chapter, and some of his suggestions are taken over, in particular the integration of linguistic constraints during generation. This integration is also a central ingredient of Stone and Webber 1998, who go one step further and argue for the simultaneous inclusion of semantic and pragmatic information as well. While both approaches look promising, it is difficult to make a precise judgement of their respective performance and predictions since central components (e.g., selection of properties, salience determination) are left unspecified. As a consequence, the computational properties of their algorithms are not clear (see Horacek 1997:212 and Stone and Webber 1998:186).

The basic idea that is pursued here is that a definite description refers to the *most salient* element satisfying the descriptive content (Lewis 1979:348-350, see Krahmer 1998 for a formalisation in dynamic semantics). Lewis mentions the following example (due to McCawley):

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

¹For a discussion of this aspect of their work, see Oberlander 1998.

Lewis notes that this statement can only be true in a domain which contains at least two dogs, which entails that *the dog* cannot be a distinguishing description. According to Lewis, (1) means that the most salient dog got in a fight with some less salient dog. Lewis does not mention descriptions which refer to ‘unique’ entities, but it is readily seen that they can also be understood in terms of salience: if there is only one entity with the relevant properties, it *has* to be the most salient one. Arguably, a notion of salience is implicit in Dale and Reiter’s usage of ‘context sets’, and also in extensions such as Horacek 1997:210 and Stone and Webber 1998:183 one can find explicit remarks that some form of salience is important. None, however, specify how salience should be determined nor which repercussions the inclusion of a notion of salience has for the generation algorithm.

In this chapter, which is partly based on Krahmer and Theune 1998 and Krahmer and Theune 1999, it is shown that it is possible to integrate an explicit notion of salience into Dale and Reiter’s Incremental Algorithm (which is described in Section 4.2) and that this paves the way for the context-sensitive generation of definite descriptions (Section 4.3). In Section 4.4, two ways of assigning salience weights to entities are discussed, one based on the hierarchical focusing constraints of Hajičová 1993, the other on the constraints of Centering Theory (Grosz et al. 1995), and a synthesis between the two is proposed. In Section 4.5, a number of extensions of the modified Incremental Algorithm are described. Section 4.6 discusses an experiment that was carried out to test some of the hypotheses underlying the modified Incremental Algorithm, and finally Section 4.7 describes the implementation of the modified Incremental Algorithm in the LGM and the accentuation of the descriptions it produces.

4.2 The Incremental Algorithm (Dale and Reiter 1995)

This section discusses the Incremental Algorithm, described in Dale and Reiter 1995, which forms the basis for the modification proposal presented in the next section. Dale and Reiter 1995 discuss four algorithms for the generation of definite descriptions, which embody four different computational interpretations of the Gricean maxims (Grice 1975). They claim that the Incremental Algorithm is the computationally most efficient algorithm of these four, and that it obeys the Gricean maxims to the same extent as human speakers do. That is, the descriptions generated by the algorithm do not always conform literally to the maxims (in particular the maxim of brevity), but deviate from them in a ‘psychologically realistic’ way.

4.2.1 Assumptions about domains

The Incremental Algorithm (and the extensions that are discussed in the rest of this chapter) can be used in any domain which meets the following criteria (Dale and Reiter 1995:254): (i) each entity in the domain is characterised by a list of attribute value pairs, or *properties*, (ii) each entity has at least an attribute type and (iii) there may be a subsumption hierarchy on the values of certain attributes. One of the values in such a hierarchy is the *basic level value* (e.g., Rosch 1978).²

Dale and Reiter 1995 use a dogs and cats domain to illustrate the workings of the Incremental Algorithm. Although this domain may not be an obvious choice for language generation applications, it is suitable for illustrative purposes, as it fits the above criteria very well: cats and dogs are familiar physical entities with easily perceivable properties, and at least one of their attribute values (*type*) can be organised in a subsumption hierarchy.³ Therefore, in this chapter Dale and Reiter are followed, sticking to the cats and dogs domain. In the current section, Dale and Reiter's format is adopted for representing the entities in the domain, although this deviates slightly from the representation as typed data structures used so far.

In the current section, the example domain is D_1 , consisting of the following four entities $d_1 - d_4$.

```

d1  < type, chihuahua >, < size, small >, < colour, black >
d2  < type, chihuahua >, < size, large >, < colour, white >
d3  < type, siamese cat >, < size, small >, < colour, black >
d4  < type, poodle >, < size, small >, < colour, white >

```

Assume that there is a subsumption hierarchy for the attribute *type*: dog subsumes chihuahua and poodle, cat subsumes siamese cat, and mammal in its turn subsumes dog and cat. Following Dale and Reiter, the basic level values for *type* are taken to be dog and cat respectively.

²Basic levels are the first levels which children learn to understand and use. Additionally, and most relevant for current purposes, basic levels are normally used in neutral contexts. Lakoff 1987:46: "For example, There's a dog on the porch *can be used in a neutral context, whereas special contexts are needed for* There's a mammal on the porch *or* There's a wire-haired terrier on the porch. (See Cruse 1977)." From the perspective of generation, the basic level values might not only be viewed as context-dependent (cf. Rosch 1978:42) but also as user-dependent (Dale, p.c.). For instance, it seems likely that for a professional dog-breeder the 'basic level' is below *dog* in the subsumption hierarchy. In this chapter, however, the basic levels are treated as given.

³A type hierarchy could in principle be constructed for the colour attribute as well, but that is not done here.

4.2.2 Outline of the Incremental Algorithm

The Incremental Algorithm is shown in Figure 4.1.⁴ The input of the algorithm is the intended referent r , the ‘contrast set’ C (i.e., the set of entities from which r has to be distinguished, consisting of all elements of the current context set except r) and, crucially, a list of preferred attributes P . This list contains all the attributes that are used in a particular domain, in order of preference. For instance, it seems likely that a human speaker would first try to describe an animal by its type attribute (is it a dog? is it a cat?), and if that does not help attributes like colour and size may be used. It is reasonable to assume that speakers have a general preference for *absolute* properties such as colour, which are easily observed without taking the other entities into account, over *relative* properties such as size, which are less easily observed and always require inspection of the distractors. (See, e.g., Pechmann 1989, Levelt 1989 and Beun and Cremers 1998.) Thus let us assume that the list of preferred attributes for the example domain is $\langle \text{type, colour, size} \rangle$. Essentially, the Incremental Algorithm goes through the list of preferred attributes, and for each attribute it encounters it finds the *best value* of this property. The best value of a property is the value closest to the basic level value such that none of the values it subsumes rule out *more* entities in the contrast set C . A value only rules out those entities of which the user knows or can easily perceive that they do not have this value. To check this, the function `UserKnows` is used. Dale and Reiter 1995 make the simplifying assumption that the user knows all the attribute values of the entities in the domain, as well as the hierarchical relationships between these values. This means that `UserKnows` ($x, \langle A, V \rangle$) returns true if and only if database entity x has value V (or a value that subsumes V) for attribute A . If the best value of a property rules out at least one distractor, the property is added to the set of properties to be used in the generation of the distinguishing description. The algorithm stops when the end of the list of preferred attributes is reached (failure), or when the set of distractors is empty (success). In the latter case, it is checked whether the type property is an element of the set of selected properties, and if not, its basic value is added to L .

⁴In this figure, some minor mistakes are corrected that occur in Dale and Reiter’s 1995 rendition of the algorithm: (i) in the third line, “ $V = \dots$ ” has been replaced with “ $V \leftarrow \dots$ ”, as this concerns an assignment, not a check; (ii) in the sixth line of the function `FindBestValue` the missing parameter r has been added to the recursive call of `FindBestValue`; (iii) in the same line, `nil` has been replaced by `novalue`, as `FindBestValue` never returns `nil`; (iv) for reasons of parallelism, `nil` has also been replaced by `novalue` in the fifth line of `FindBestValue`; (v) `{}` is consistently used instead of `nil` to denote the empty set in the output of `RulesOut`.

```

MakeReferringExpression ( $r, C, P$ )
 $L \leftarrow \{\}$ 
for each member  $A_i$  of list  $P$  do
   $V \leftarrow \text{FindBestValue}(r, A_i, \text{BasicLevelValue}(r, A_i))$ 
  if  $\text{RulesOut}(\langle A_i, V \rangle) \neq \{\}$ 
  then  $L \leftarrow L \cup \{\langle A_i, V \rangle\}$ 
     $C \leftarrow C - \text{RulesOut}(\langle A_i, V \rangle)$ 
  endif
  if  $C = \{\}$ 
  then if  $\langle \text{type}, X \rangle \in L$  for some  $X$ 
    then return  $L$ 
    else return  $L \cup \{\langle \text{type}, \text{BasicLevelValue}(r, \text{type}) \rangle\}$ 
  endif
endif
return failure

FindBestValue ( $r, A, \text{initial-value}$ )
if  $\text{UserKnows}(r, \langle A, \text{initial-value} \rangle) = \text{true}$ 
then  $\text{value} \leftarrow \text{initial-value}$ 
else  $\text{value} \leftarrow \text{novalue}$ 
endif
if ( $\text{more-specific-value} \leftarrow \text{MoreSpecificValue}(r, A, \text{value}) \neq \text{novalue} \wedge$ 
  ( $\text{new-value} \leftarrow \text{FindBestValue}(r, A, \text{more-specific-value}) \neq \text{novalue} \wedge$ 
  ( $|\text{RulesOut}(\langle A, \text{new-value} \rangle)| > |\text{RulesOut}(\langle A, \text{value} \rangle)|$ ))
then  $\text{value} \leftarrow \text{new-value}$ 
endif
return  $\text{value}$ 

RulesOut ( $\langle A, V \rangle$ )
if  $V = \text{novalue}$ 
then return  $\{\}$ 
else return  $\{x : x \in C \wedge \text{UserKnows}(x, \langle A, V \rangle) = \text{false}\}$ 
endif

```

FIGURE 4.1: The Incremental Algorithm of Dale and Reiter 1995.

4.2.3 Example

Consider domain D_1 , and suppose that the context set contains all entities in the domain.⁵ Now we want to generate an expression for d_2 , using the Incremental Algorithm. The contrast set C of d_2 is $\{d_1, d_3, d_4\}$ (i.e., all elements in D_1 except d_2), and the list of preferred properties P is assumed to be $\langle \text{type, colour, size} \rangle$. $\text{MakeReferringExpression}(d_2, C, P)$ is called, and the set of selected properties L is initialised as $\{\}$.

The first property of d_2 that is considered by the Incremental Algorithm is $\langle \text{type, chihuahua} \rangle$. The function FindBestValue is called for this property, with dog (the basic level value for d_2 's type attribute) as the initial value. FindBestValue first tries to find a more specific value for this attribute, which is chihuahua . Now, FindBestValue is recursively called with chihuahua as the initial value. Since no more specific value can be found, this recursive call returns chihuahua . This new value rules out both d_3 and d_4 , whereas the basic value dog only rules out d_3 . As a consequence, chihuahua is returned as the best value of d_2 's type attribute. Since $\text{RulesOut}(\langle \text{type, chihuahua} \rangle) \neq \{\}$, this property has sufficient descriptive content to be included in the description under construction. It is added to L and the set of distractors C is reduced with d_3 and d_4 . This does not leave C empty (not all distractors are ruled out), and so the algorithm proceeds by taking the second property of d_2 , $\langle \text{colour, white} \rangle$. Since there is no subsumption hierarchy for the values of colour, there is only one level and white is by definition the best value for this attribute. The property $\langle \text{colour, white} \rangle$ rules out the remaining distractor d_1 , which is black. Consequently, this property is added to L and d_1 is removed from C . Since C is now empty, and L already contains the type of d_2 , the selection of properties is finished and the set $\{\langle \text{type, chihuahua} \rangle, \langle \text{colour, white} \rangle\}$ is returned.

4.2.4 Discussion

One of the central features of the Incremental Algorithm is that there is no backtracking (hence the term 'incremental'): once a property p has been selected, it will be realised in the final description, even if a property which is added later would render the inclusion of p redundant with hindsight. This makes the algorithm very efficient. Dale and Reiter (1995:247) argue that the Incremental Algorithm has a polynomial complexity and the theoretical run time can be characterised as $n_d n_l$: the run time depends solely on the number of distractors n_d and the number of

⁵Dale and Reiter 1995 do not specify how the contents of a context set should be determined. In all their examples, the context set is equal to the set of all entities in the domain.

iterations (i.e., selected properties) n_i . This means that the Incremental Algorithm is the fastest algorithm discussed in Dale and Reiter 1995.

Dale and Reiter additionally claim that the algorithm's lack of backtracking is 'psychologically realistic' since human speakers also often include redundant modifiers in their referring expressions (see e.g., Pechmann 1989). They write (Dale and Reiter 1995:248): "*For example, in a typical experiment a participant is shown a picture containing a white bird, a black cup, and a white cup and is asked to identify the white bird; in such cases, participants generally produce the referring expression the white bird, even though the simpler form the bird would have been sufficient.*" Yet, it seems that the Incremental Algorithm would produce the description *the bird* in this situation: starting from the natural assumption that type is the most preferred attribute, the property $\langle \text{type, bird} \rangle$ will be the first one selected and immediately rules out the black and the white cup.

It should be noted that the notion of incrementality generally refers to *speech production*; the fact that speakers, when describing an entity, start uttering properties of that entity without making sure whether these are actually distinctive or not. Dale and Reiter's incrementality refers to the lack of backtracking in property selection, but the order in which properties are selected is not related to the order in which properties are realised in speech. Full incrementality in the latter sense (each property is uttered as soon as it is selected), cannot be obtained without taking a certain amount of lookahead into account (Levelt 1989).

In developing the Incremental Algorithm, Dale and Reiter mainly aimed at achieving computational efficiency, while at the same time allowing human preferences and capabilities to be taken into consideration. As a consequence, various other aspects of definite description generation have remained somewhat underdeveloped. Some of these aspects are further worked out in this chapter. For instance, Dale and Reiter 1995 do not specify how the linguistic context influences the selection of properties to be included in a definite description. This issue, which is particularly important for the generation of anaphoric descriptions, is central to the modifications of the Incremental Algorithm proposed in this chapter. In addition, Dale and Reiter 1995 only discuss the generation of descriptions of the form "the [Adj] N". Section 4.5 describes how pronouns, relational and bridging descriptions can be generated using a modified version of the Incremental Algorithm. Finally, following Horacek 1997, the algorithm is extended with a check on the expressibility of selected properties and a procedure to build a natural language expression within the algorithm.

There are several other issues which are not dealt with by Dale and

Reiter 1995, and which are not addressed here either. One example is the treatment of relative attribute values. Following Dale and Reiter, all attribute values are treated here as if they are absolute, assuming that relative attribute values such as *big* are simply given in the domain database. In addition, as noted in Horacek 1995, the Incremental Algorithm does not take prominence of properties into account. This issue is dealt with in Horacek 1995 and Horacek 1997 by always including a prominent property of the intended referent, even if it does not rule out any distractors (an example might be including the colour when describing a pink elephant in a group of flamingoes). The current chapter only discusses prominence (or *salience*) of discourse entities, not of properties.

4.3 A modification of the algorithm based on salience

This section discusses a modification of the Incremental Algorithm based on salience, which takes the linguistic context into account during the generation of definite descriptions. After the modification has been motivated and the basic ideas underlying it have been sketched, an outline of the modified algorithm is given and illustrated using some examples.

4.3.1 Motivation: determining the context set

The contents of the descriptions generated by the Incremental Algorithm are to a large extent determined by the context set. Nevertheless, Dale and Reiter do not address the question how such context sets are constructed nor how their contents can be updated during the generation process. They only write:

We define the context set to be the set of entities that the hearer is currently assumed to be attending to; this is similar to the set of entities in the focus spaces of the discourse focus stack in Grosz and Sidner's 1986 theory of discourse structure. (Dale and Reiter 1995:236)

Dale 1992:192-193 is somewhat more explicit and briefly describes a full and a partial order on focus spaces. However, he concludes that:

[i]n the present domain [recipes, MT], 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.

(The 'working set' refers to the set of "identifiable distinct entities in the domain at any point in time," Dale 1992:56.) The aim of this chapter is to be explicit about the continuously changing contents of context sets and the repercussions this has for the Incremental Algorithm. It is instructive at this point to look at some of the available options. First of all, there always has to be a domain of discourse D : the total set of

entities which can be referred to (in a data-to-speech system this set is given as part of the data). Some people have suggested that a context set may be a proper subset of D , containing those entities of D which have been referred to before, or which are prominent for some other reason. This should make it possible to generate reduced anaphoric descriptions, because the intended referent only has to be distinguished from members of the context set, and not from all domain entities. However, there appear to be some problems with this approach. This can be illustrated using the following example.

Suppose a text must be generated about the visit of two sisters, Alice and Karen, to a dog show. Thus, the domain of discourse contains these two plus all dogs that are present at the dog show, i.e., hundreds of dogs of all kinds, sizes, and colours. When generation starts, the context set is assumed to equal the domain of discourse, since at the beginning of the discourse all entities in the domain are equally prominent (or, in terms of Grosz and Sidner 1986, the focus space is still empty).⁶ If, in this situation, a distinguishing description must be generated for one of the dogs, say d_{45} , this description will necessarily include many of its properties, e.g., *the large black long-haired sausage dog*. This is illustrated in sentence (2)a (repeated as (3)-(5)a).

After d_{45} has been referred to, there are two possibilities with respect to the context set. The first option is to reduce the context set, following the approach outlined above. Because Alice and d_{45} have been mentioned, they are added to the focus space, and therefore the context set may be restricted to include only these two entities. This step makes it possible to generate a reduced anaphoric description of d_{45} : when this entity is referred to for a second time, as in (2)b, the Incremental Algorithm will produce the description *the dog*, as it is the only dog in the current context set.

Restricting the context set:

- (2) a. Alice liked the₄₅ large black long-haired sausage dog.
 b. Karen liked the₄₅ dog too.
- (3) a. Alice liked the₄₅ large black long-haired sausage dog.
 b. ?? Karen preferred the₅₃ grey dog.

⁶This is a simplifying assumption, which has no influence on the arguments presented here. In reality, it will often be the case that some entities in the domain are initially more salient than others, for instance because they are (physically) closer to the hearer.

However, an unwanted consequence of restricting the context set is that the descriptions of *all* domain entities will be made relative to this restricted set. That this is problematic, is illustrated by example (3). Sentence (3)a is identical to (2)a, but (3)b contains a reference to another entity from the domain, d_{53} , which happens to be a small grey pygmy poodle with a perm wave. As the reader may check, given the restricted context set the Incremental Algorithm describes d_{53} as *the grey dog*, even if there are various grey dogs in the domain of discourse. Clearly, this description is inadequate: although it does distinguish d_{53} from the entities in the restricted context set, it will not enable the hearer to single out the intended referent from all dogs present at the dog show. This example makes it clear that when d_{53} is introduced to the discourse, it must be distinguished from all entities in the domain, and not only from those in the context set. Therefore, the context set cannot be restricted to contain only those entities that have been mentioned.

The other option is not to restrict the context set. This allows for an adequate initial description of d_{53} , as shown in (5)b, but it causes a problem with the anaphoric reference to d_{45} in (4)b. Since the context set has not changed, the reference to d_{45} in (4)b is again made relative to the entire domain, and this causes a repetition of the initial description *the large black long-haired sausage dog*, which is clearly undesirable.

Not restricting the context set:

- (4) a. Alice liked the₄₅ large black long-haired sausage dog.
- b. ?? Karen liked the₄₅ large black long-haired sausage dog too.

- (5) a. Alice liked the₄₅ large black long-haired sausage dog.
- b. Karen preferred the₅₃ small grey pygmy poodle with the perm wave.

In sum, restricting the context set to a proper subset of the domain, and then generating all distinguishing descriptions relative to this set (minus the intended referent) does not always produce the desired results. Apparently, different context sets should be used depending on the entity to be described. An entity that is being newly introduced to the discourse must be distinguished from all other entities in the domain, whereas an entity that has been previously mentioned can have a reduced description. The proposed solution to this problem is to structure the domain by marking certain entities as more prominent than others. Descriptions are then required to be distinguishing relative to the set of entities which are equally or more prominent than the intended referent. As our metric of prominence *salience weights* shall be used.

4.3.2 Definite descriptions and salience

The underlying idea of the proposed modifications to the Incremental Algorithm is the following:

A definite description “the \bar{N} ” is a suitable description of an entity d in a state s iff d is the most salient entity with the property expressed by \bar{N} in state s .

So, for example, the description *the dog* is suitable if it is used to refer to the currently most salient dog (like d_{45} in (2)b). To model the salience of an entity, a function sw (salience weight) is used, which given a state s and an entity d yields a natural number. After a sentence has been generated, a new state is entered. For the sake of simplicity, it is assumed that in the initial state (s_0), say the beginning of the generation process, all entities are minimally salient. Formally, $\forall d \in D : sw(s_0, d) = 0$. When an entity is referred to during generation, its salience weight increases. Conversely, when an entity has not been mentioned for some time, its salience weight may decrease (but not below zero, which is the minimal salience level). Section 4.4 discusses and compares two methods for salience weight assignment, which specify under which circumstances a given entity increases or decreases in salience, and how much. For the time being, salience weights are simply assumed to be given.

The use of a point-wise salience weight assignment is proposed here, since this allows for fine distinctions in salience which cannot be made in a group-wise assignment (for instance in terms of focus spaces). Point-wise assignment is in line with the intuition that salience is not an all-or-nothing phenomenon, but that the salience of certain entities may gradually change with the flow of the discourse. Section 4.4 discusses some examples where the additional information provided by point-wise assignments is put to use, and Section 4.5.1 shows that this information is also potentially useful for the sake of pronominalisation. Finally, it should be noted that a point-wise assignment can always be mapped onto a group-wise assignment (e.g., by regarding all entities whose salience weight is zero as ‘non-salient’ and all others as ‘salient’), but not vice versa.

An entity r is defined to be the most salient entity having certain properties L in state s (notation: $\text{MostSalient}(r, L, s)$) if, and only if, there are no entities in the domain that are known to have properties L and that have a salience weight which is higher or equal to the salience weight of r in s . Notice that r only has to be distinguished from those entities that have a salience weight which is higher or equal to the salience weight of r in s , so the set of *distractors* of r in s can be defined as in Definition 1 below.

Definition 1 (Distractor set)

$$\text{Dist}(r, s) = \{d : d \in D \wedge d \neq r \wedge sw(s, d) \geq sw(s, r)\}$$

$\text{MostSalient}(r, L, s)$ can also be defined in terms of this distractor set: an entity r is the most salient entity having certain properties L in state s if, and only if, all entities in its distractor set are ruled out by the properties in L . Put differently: the set of entities from $\text{Dist}(r, s)$ that are ruled out by L is equal to $\text{Dist}(r, s)$ itself. The RulesOut function used here is the same as the one used in the Incremental Algorithm, except that the set of distractors is explicitly given as a parameter. (See Figure 4.2.) Note that the set of entities ruled out by a *set* of properties is equal to the union of the sets of entities ruled out by each of the properties in the set.

Definition 2 (Salience condition)

$$\text{MostSalient}(r, L, s) \Leftrightarrow \text{RulesOut}(\text{Dist}(r, s), L) = \text{Dist}(r, s)$$

A crucial difference between the current set of distractors and Dale and Reiter's contrast set C is that the former is defined relative to the entity to be described. Entities that have a lower salience weight than r in s are not included in the distractor set of r in s . This means that the distractor set of an entity which has a zero salience weight (for instance because it has not yet been mentioned) is equal to the set of all elements in the domain of discourse D , and that the set of distractors for an entity which has a non-zero salience weight may be a proper subset of D (the set may even be empty, if the intended referent is the single most salient entity in the domain). The advantage of this approach can be illustrated using the earlier dog show example. Assume that d_{45} , but no other dog, has just been mentioned. If d_{45} is then referred to for a second time, its distractor set is empty, since d_{45} is more salient than all other dogs in the domain. This means that it is sufficient to describe it as *the dog*, as in (2)b. On the other hand, if d_{53} is mentioned for the first time in the same situation, the distractor set for the latter dog consists of all dogs at the dog show (despite the recent mention of d_{45}). This means that a full description of d_{53} is required, as in (5)b. Some more worked out examples are given below in Section 4.3.4.

4.3.3 Outline of the modified algorithm

Figure 4.2 contains the proposal for a modified algorithm in pseudo-code. The main differences between the original Incremental Algorithm and the modified algorithm are that the latter (i) defines success in terms of salience, and (ii) uses a distractor set Dist , which – unlike Dale and Reiter's contrast set C – is constructed relative to the entity to be described, taking salience weights into account. A third difference is that

```

MakeReferringExpression ( $r, s$ )
 $P \leftarrow$  PreferredAttributes (Datatype ( $r$ ))
 $RemDist \leftarrow$  Dist ( $r, s$ )
 $tree \leftarrow$  nil
for each member  $A_i$  of list  $P$  do
   $V \leftarrow$  FindBestValue ( $r, A_i, \text{BasicLevelValue} (r, A_i), RemDist$ )
   $OutRuledRefs \leftarrow$  RulesOut ( $RemDist, \langle A_i, V \rangle$ )
   $tree' \leftarrow$  UpdateTree ( $tree, \langle A_i, V \rangle$ )
  if ( $OutRuledRefs \neq \{\}$   $\vee A_i = \text{type}$ )  $\wedge tree' \neq \text{nil}$ 
    then  $tree \leftarrow tree'$ 
       $RemDist \leftarrow RemDist - OutRuledRefs$ 
  endif
  if  $RemDist = \{\}$  %  $r$  is most salient given  $L$  in  $s$ 
    then  $tree \leftarrow$  AddDefDet ( $tree$ )
      return  $tree$ 
  endif
endfor
 $tree \leftarrow$  AddIndefDet ( $tree$ )
return  $tree$ 

Dist ( $r, s$ )
return  $\{d : d \in D \wedge d \neq r \wedge sw(s, d) \geq sw(s, r)\}$ 

FindBestValue ( $r, A, \text{initial-value}, RemDist$ )
if UserKnows ( $r, \langle A, \text{initial-value} \rangle$ ) = true
  then  $value \leftarrow \text{initial-value}$ 
else  $value \leftarrow \text{novalue}$ 
endif
if ( $msv \leftarrow \text{MoreSpecificValue} (r, A, value) \neq \text{novalue} \wedge$ 
  ( $new-value \leftarrow \text{FindBestValue} (r, A, msv, RemDist) \neq \text{novalue} \wedge$ 
   $|\text{RulesOut} (RemDist, \langle A, new-value \rangle)| > |\text{RulesOut} (RemDist, \langle A, value \rangle)|$ )
  then  $value \leftarrow new-value$ 
endif
return  $value$ 

RulesOut ( $RemDist, \langle A, V \rangle$ )
if  $V = \text{novalue}$ 
  then return  $\{\}$ 
else return  $\{x : x \in RemDist \wedge \text{UserKnows} (x, \langle A, V \rangle) = \text{false}\}$ 
endif

```

FIGURE 4.2: Full sketch of the modified algorithm.

the modified algorithm does not return a set of properties to be included in a description. Instead, it returns the syntactic tree for the description itself, which is constructed during generation. This way, linguistic restrictions (e.g., a limitation on the number of relative clauses included in the description) can be taken into account. (See Horacek 1997.)

In addition, a few relatively minor improvements have been made to the original Incremental Algorithm. The first of these modifications involves the list of preferred properties P . Dale and Reiter 1995 give this list as an input parameter of the algorithm, leaving it implicit that P depends on the type of the referent to be described. In the modified algorithm, P is not a parameter of unknown origin, but a variable that is initialised at the start of the algorithm, relative to the data type of the intended referent r . In addition, the modified algorithm never returns ‘failure’; it always returns a description for the intended entity, even if it has not been able to generate a distinguishing description. When the algorithm has iterated through the entire list of preferred properties without success, it returns an *indefinite* expression containing the selected properties. The idea behind this is that it is better to generate a non-distinguishing description for an entity than no description at all.⁷ A final difference between the modified algorithm and the Incremental Algorithm is that, after having determined the best value for the type attribute, the modified algorithm immediately adds this value to the set of selected properties, regardless of its discriminatory power. This is more efficient than adding its basic level value afterwards, and it gives the same results, because when type has no discriminatory power, its best value is always the basic level value.⁸

These differences aside, to ease comparison the modified algorithm sticks as closely as possible to the algorithm from Dale and Reiter 1995: 257. The adopted notation is the one employed by Dale and Reiter, which is similar to the WEB style from Knuth 1986, viiff. Below, the modified algorithm is illustrated with a number of examples. First, a general, somewhat informal overview is given.

The algorithm `MakeReferringExpression` (r, s) tries to generate a definite description for a referent r in a state s . The following variables are used: P is the list of *preferred attributes*, initialised by means of the

⁷Instead of always producing an indefinite description of the form *an X*, as is proposed here, in certain contexts it might be better to generate *one of the Xs*, as suggested in Dale and Reiter 1995. This issue is left for future investigation.

⁸Note that it is assumed here that the `type` attribute is always the first element of the list of preferred attributes, and that it can always be expressed; if this were not the case, in the current version of the algorithm an incomplete tree would be returned if all distractors were ruled out before the `type` attribute was encountered.

function `PreferredAttributes` which, given a referent r of a specific data type, returns the list of preferred attributes for this data type; $RemDist$ is the set of *remaining* distractors of r in s , initialised as $Dist(r, s)$; and $tree$ contains the syntactic tree for the NP under construction, expressing the selected properties of r . In the modified algorithm, $tree$ replaces the set of properties L used by Dale and Reiter 1995.

As in the original version of the algorithm, the main loop iterates through the list P of preferred attributes. For each attribute A on this list, the best value V is sought. This is done essentially in the same way as in the Incremental Algorithm, except that it is done relative to the set of remaining distractors of r , i.e., the best value is the one that rules out most members of the set $RemDist$. Once the best value V is found, the variable $OutRuledRefs$ is instantiated as the set of remaining distractors ruled out by the property $\langle A, V \rangle$, and the variable $tree'$ is instantiated as the tree that results from updating $tree$ (the tree expressing the selected properties of r) with an expression for $\langle A, V \rangle$. If $tree$ cannot be updated successfully because the lexical or syntactic restrictions of the generation module make it impossible to express the property $\langle A, V \rangle$ (see below), `UpdateTree($tree, \langle A, V \rangle$)` returns nil. If the property $\langle A, V \rangle$ actually rules out any of the remaining distractors ($OutRuledRefs \neq \{\}$), or if it is the type property (which is always selected), and the tree under construction can be updated successfully ($tree' \neq nil$), it is selected for inclusion. The variable $tree$ is instantiated as $tree'$ (its updated version) and the remaining distractors ruled out by the selected property ($OutRuledRefs$) are removed from $RemDist$. The algorithm then checks if the set of remaining distractors is empty. If this is the case, the selection of properties is finished, because r is now the *most salient* entity with the selected properties in the current state of the discourse. Note that the Saliency condition (Definition 2) does not occur in its literal form in the algorithm, because if $RemDist$ is empty, this means that all distractors of r in s are ruled out by the selected properties, which is equivalent to the more complex Definition 2. Then the function `AddDefDet` inserts the determiner *the* in the tree under construction and returns a full definite description expressing the selected properties. If the modified algorithm has exhausted the list of preferred properties without returning a distinguishing description, it inserts the indefinite determiner $a(n)$ to the tree and returns the resulting indefinite description.

In contrast with the rest of the algorithm, the function `UpdateTree` is largely domain and language dependent. Since this function is not the focus of this research, it is not specified in detail here. The following is a broad sketch of the function, based on a few simplifying assumptions. Roughly, `UpdateTree` works as follows: starting from a prototypical NP

structure, the function attempts to integrate each new value V in the syntactic tree constructed so far. The value of the `type` attribute (and no other value) is assumed to be always realised as the head noun, and therefore can always be included in the tree under construction. Unary properties are added as prenominal AP modifiers to designated slots in the tree. (For a discussion of adjective orderings, see Dale 1992:127-130, and the references cited therein.) Relations (discussed in Section 4.5.2) may be realised as postnominal PPs or relative clauses. Following Horacek 1997, each available slot in the prototypical NP structure is allowed to be filled only once. This way, the generation of awkward descriptions such as the one in (6) can be avoided.

(6) ?? The poodle with the perm wave with the black collar ...

No lexical selection takes place: each attribute-value pair is assumed to be associated with a fixed lexicalised tree; e.g., $[_{AP} \textit{small}]$ for $\langle \textit{size}, \textit{small} \rangle$. An example of tree construction is given in Section 4.3.4.

4.3.4 Examples

In the examples discussed in this section, domain D_1 from Section 4.2.1 is taken as the domain of discourse. Below, in Figure 4.3, domain D_1 is shown using the same kind of feature structure representation as was used in the previous chapters, in line with the use of typed data structures in the LGM. All entities in the domain are assumed to be data structures of type *animal*, which has the fields `referent`, `type`, `size`, and `colour`. The preferred list of attributes P for entities of data type *animal* is assumed to be $\langle \textit{type}, \textit{colour}, \textit{size} \rangle$.

In the first two examples below, the generation of non-anaphoric descriptions is briefly illustrated, showing that (i) when all entities in the domain are equally salient, the modified algorithm selects exactly the same properties as the original Incremental Algorithm, and (ii) the

$\left[\begin{array}{l} \textit{animal} \\ \textit{referent} : d_1 \\ \textit{type} : \textit{chihuahua} \\ \textit{size} : \textit{small} \\ \textit{colour} : \textit{black} \end{array} \right]$	$\left[\begin{array}{l} \textit{animal} \\ \textit{referent} : d_2 \\ \textit{type} : \textit{chihuahua} \\ \textit{size} : \textit{large} \\ \textit{colour} : \textit{white} \end{array} \right]$
$\left[\begin{array}{l} \textit{animal} \\ \textit{referent} : d_3 \\ \textit{type} : \textit{siamese cat} \\ \textit{size} : \textit{small} \\ \textit{colour} : \textit{black} \end{array} \right]$	$\left[\begin{array}{l} \textit{animal} \\ \textit{referent} : d_4 \\ \textit{type} : \textit{poodle} \\ \textit{size} : \textit{small} \\ \textit{colour} : \textit{white} \end{array} \right]$

FIGURE 4.3: Domain D_1 , represented as a set of data structures.

modified algorithm makes fully explicit from which other entities in the domain the intended referent should be distinguished. The modified algorithm is illustrated in more detail in the two final examples, which deal with the generation of reduced anaphoric descriptions that are only distinguishing in context.

In the examples given below, a very basic form of salience weight assignment is used, simply assuming that entities become more salient after having been mentioned. This assumption is sufficient to deal with the short examples discussed here. Section 4.4 introduces two principled methods for salience weight assignment and discusses a synthesis of those, based on more elaborate examples that require a more complex form of salience weight assignment.

Non-anaphoric description (1).

Suppose that an expression for d_2 must be generated in the initial state s_0 , at the beginning of the discourse. (This corresponds to the example given in Section 4.2.3 to illustrate the working of the Incremental Algorithm.) Because in s_0 all entities in D_1 are equally salient by assumption, the distractor set Dist contains the same entities as Dale and Reiter's contrast set C in the earlier version of this example (i.e., it contains all entities in D_1 except d_2 itself). In this situation, the modified algorithm selects the same properties as the Incremental Algorithm, returning a syntactic tree for the description *the white chihuahua*.

Non-anaphoric description (2).

Suppose that after having generated a discourse-initial sentence containing a reference to d_2 (e.g., *The white chihuahua was chewing a bone*), a second sentence must be generated that contains a reference to d_1 . As discussed in Section 4.3.1, in the Incremental Algorithm it is not clear whether in this situation the context set should be reduced to contain only d_2 (the only entity that has been mentioned so far) or whether it should remain equal to the set of all entities. The former option is not attractive, as it would cause the Incremental Algorithm to describe d_1 as *the black dog* (which does not distinguish it from d_4), and the latter option is not attractive either as it leaves no room for a reduced description of d_2 later on.

In the modified algorithm, there is no problem determining the set of entities from which d_1 must be distinguished. After having generated a description for d_2 in s_0 , a new state s_1 is entered, where d_2 is more salient than all other entities in the domain. Although d_2 is currently the most salient entity, the distractor set of d_1 still contains all entities in D_1 except d_1 itself: $\text{Dist}(d_1, s_1) = \{d_2, d_3, d_4\}$, since $sw(d_1, s_1) = 0$ and

all other entities in the domain have a salience weight which is higher than or equal to zero. In order to distinguish d_1 from the entities in this set, the properties $\langle \text{type, chihuahua} \rangle$ (ruling out d_3 and d_4) and $\langle \text{colour, black} \rangle$ (ruling out d_2) are selected, and the syntactic tree for *the black chihuahua* is returned. In (7), this description is embedded in an example text.

- (7) The₂ white chihuahua was chewing a bone. The₁ black chihuahua came a little too close.

Anaphoric description (1).

Anaphoric descriptions generally contain less information than their antecedents. This may be reflected by the omission of properties, but also by the use of a more general head noun. The latter case is illustrated by the following example, concerning the generation of an anaphoric description for d_2 in state s_2 , which is entered after text (7) has been generated. In this state, d_1 and d_2 are assumed to be equally salient, and more salient than the other entities in D_1 .

When calling `MakeReferringExpression` (d_2, s_2), the list P is initialised as `PreferredAttributes` (*animal*) = $\langle \text{type, colour, size} \rangle$. The set of remaining distractors *RemDist* is initialised as $\text{Dist}(d_2, s_2) = \{d_1\}$: d_1 is the only entity in the domain other than d_2 that has a salience weight that is equal to or greater than that of d_2 in s_2 . In addition, *tree* is initialised as nil. The first property considered by the algorithm is *type*. The best value for this attribute is the basic level value *dog*, since *dog* rules out the same number of distractors as the more specific value *chihuahua*: both *chihuahua* and *dog* rule out 0 distractors. This means that *OutRuledRefs* is instantiated as $\{\}$. The function `UpdateTree` then incorporates an expression for *dog* (i.e., the word *dog*) in the tree under construction (which is currently still empty). As said, the *type* value is assumed to be always realised as the head noun. The result, shown as tree (I) from Figure 4.4, is stored in *tree'*. Because the *type* property is always included, even if it rules out no distractors, and the update has been successful (*tree'* is not nil), the property $\langle \text{type, dog} \rangle$ is selected for inclusion in the description and *tree* is instantiated as *tree'*. No entities are removed from *RemDist*, and because *RemDist* is not empty, the algorithm proceeds by taking the second property of d_2 , $\langle \text{colour, white} \rangle$. This property does rule out d_1 , so the function `UpdateTree` adds an AP for the colour *white* (lexicalised as *white*) to the current NP tree (giving tree II in Figure 4.4). Distractor d_1 is removed from *RemDist*, leaving the set empty: d_2 is the most salient white dog in the domain. Finally, the definite article *the* is added to the constructed NP tree, and the

4.3.5 Discussion

In this section, a generalisation of the Incremental Algorithm has been described which extends the original in a number of ways. First and foremost, it explicitly takes the linguistic context into account by treating context sets as a combination of a discourse domain with a salience function. This generalisation entails that creating a referring expression for one entity (here d_2 of domain D_1) can result in e.g., *the white chihuahua* (in the initial state), *the white dog* (in a context like *The white chihuahua and the black chihuahua ...*) and *the chihuahua* (sample context: *The white chihuahua and the poodle ...*).

This does not make the algorithm more complex: it still has a polynomial complexity, and a similar theoretical run time as the original algorithm ($n_d n_l$). To see this, observe that the modified algorithm still requires as many iterations as properties realised in the description (n_l) and in each of the iterations has to inspect the set of distractors (n_d).

Because the focus has been on adding salience to the Incremental Algorithm, its modified version still exhibits some of the limitations of the original algorithm, discussed in Section 4.2.4. Following Dale and Reiter 1995, simplifying assumptions have been made about relative attribute values such as big and small, and issues like the treatment of prominent properties (see Horacek 1997) or the generation of plurals have not been dealt with. These issues are left for future work.

4.4 Determining salience weights

So far it has not been discussed how salience weights are computed. The literature contains various methods to do so, such as Alshawi 1987, Hajičová 1993, Lappin and Leass 1994, and Grosz et al. 1995. The basic idea underlying these methods is the same: entities that have (recently) been mentioned are more salient than entities that have not been mentioned. This basic idea is worked out differently for each method, for instance in the way that factors like syntactic position or discourse function are taken into account. As a consequence, the choice for a specific method of salience weight assignment may influence the outcome of the algorithm. To illustrate this, the hierarchical focusing constraints of Hajičová 1993 and the Centering approach of Grosz et al. 1995 are discussed here. Throughout this section, sw is a total function mapping the entities in a domain D to the set $\{0, \dots, 10\}$, with the intuition that 0 represents complete non-salience and 10 maximal salience.⁹ In the

⁹It is not essential to have an 11-point scale of salience. In principle, any finite subset of integers will do.

initial situation, all entities in the domain are assumed to be minimally salient ($\forall d \in D : sw(s_0, d) = 0$).

4.4.1 Hajičová: hierarchical focus constraints

Salience weight assignment in the Praguian approach is based on the notions *topic* and *focus* (see e.g., Sgall et al. 1986). Informally speaking, the topic of utterance U is what U is about (generally this corresponds to the grammatical subject), while the focus¹⁰ of U is what the sentence says about the topic (generally this corresponds to the rest of the sentence). Informally, the Praguian approach says that entities which have been referred to in the focus part of a sentence have maximal salience; they constitute new information that is assumed to be in the centre of attention. Entities which have been referred to in the topic have a lower salience weight than those referred to in the focus. Their exact weight is determined by the form of the referring expression that has been used: if a definite description has been used, the highest-but-one salience weight is assigned, and if a pronoun has been used, the salience weight of the entity does not change. If an entity has not been referred to in a sentence (i.e., it is part of neither focus nor topic) then its salience weight decreases. The salience weight of entities that have been previously referred to in the topic part of a sentence decreases at a slower rate than that of entities that have been previously referred to in the focus. Formally:

Definition 3 (Salience weight assignment based on Hajičová 1993)

Let U_i be a sentence uttered in state s_i , and let $\text{topic}(U_i) \subseteq D$ and $\text{focus}(U_i) \subseteq D$ be the sets of entities which are referred to in the topic and the focus part of U_i respectively. Then the salience weight of entities in s_{i+1} is determined as follows:

$$sw(s_0, d) = 0 \text{ for all } d \in D$$

$$sw(s_{i+1}, d) = \begin{cases} 10 & \text{if } d \in \text{focus}(U_i) \\ 9 & \text{if } d \in \text{topic}(U_i), \\ & \text{and } d \text{ is referred to by a definite NP} \\ sw(s_i, d) & \text{if } d \in \text{topic}(U_i), \\ & \text{and } d \text{ is referred to by a pronoun} \\ \max(0, sw(s_i, d) - 1) & \text{if } d \notin \text{topic}(U_i) \cup \text{focus}(U_i), \\ & \text{and } d \in \text{topic}(U_j), j < i \\ \max(0, sw(s_i, d) - 2) & \text{if } d \notin \text{topic}(U_i) \cup \text{focus}(U_i), \\ & \text{and } d \in \text{focus}(U_j), j < i \end{cases}$$

¹⁰Notice that the notion of focus used here is very different from that used in e.g., Grosz and Sidner 1986 and Grosz et al. 1995.

The salience function given in Definition 3 above closely corresponds to the rules given by Hajičová 1993, except that in the new version entities are mapped to a *higher* number as they become more salient, with a maximum of 10, whereas in Hajičová 1993 entities are assigned a *lower* number as they become more salient, maximal salience being indicated by 0. As a consequence, the Praguian approach allows for an infinite decrease in salience weight, keeping track of entities which have faded from the discourse long ago. This seems to be neither efficient nor psychologically realistic, so here it is assumed that there is not only a maximal but also a minimal salience level: in the version given here, the salience weight of an entity cannot decrease below zero.

4.4.2 Grosz et al.: Centering

An alternative definition of salience weight assignment can be based on the ranking of the so-called *forward looking centers* (C_f) of an utterance. According to Centering Theory (Grosz et al. 1995), the set of forward looking centers of an utterance contains the entities referred to in that utterance. This set is partially ordered to reflect the relative prominence of the referring expressions within the utterance. Grammatical roles are a major factor here, so that *subject* > *object* > *other*.¹¹

Definition 4 (Salience weight assignment based on Centering Theory) Let U_i be a sentence uttered in state s_i , in which reference is made to $\{d_1, \dots, d_n\} \subseteq D$. Let $C_f(U_i)$ (the forward looking centers of U_i) be a partial order defined over $\{d_1, \dots, d_n\} \subseteq D$. Then the salience weight of entities in s_{i+1} is determined as follows:

$$sw(s_0, d) = 0 \text{ for all } d \in D$$

$$sw(s_{i+1}, d) = \begin{cases} 0 & \text{if } d \notin C_f(U_i) \\ n & \text{otherwise, where } n = \text{level}(d, C_f(U_i)) \end{cases}$$

Here, $\text{level}(d, C_f(U_i))$ refers to the level of the occurrence of d in the ordering $C_f(U_i)$, defined in such a way that the highest element(s) on the ordering are mapped to 10, the element(s) immediately below are mapped to 9, etc. Thus, $\text{level}(d_i, \langle d_1, \dots, d_n \rangle) = \max(0, 11 - i)$, where $\langle d_1, \dots, d_n \rangle$ is the ordered set of forward looking centers of the relevant utterance.

¹¹It should be noted that this ordering may differ across languages, and that features other than grammatical roles seem to be relevant as well (see, e.g., Cote 1998:57-59 for a discussion of this issue). In the current section, the 'traditional' C_f ordering is adhered to, which appears to hold (at least in simple sentences) for both English and Dutch.

4.4.3 Examples, predictions and comparison

Here two illustrative examples are discussed in some detail, with an emphasis on the repercussions of choosing either way of assigning salience weights for the generation of referring expressions. The function sw^h is used to indicate the salience weights as assigned by the Hajičová-based approach (Definition 3), while sw^c gives the salience weights as based on the Centering approach (Definition 4). Indices on words refer to entities in the domain. In the example sentences, the Praguian topic always coincides with the syntactic subject. Suppose the domain of discourse is D_1 , and consider the following example.

(10) a. The₂ white chihuahua was angry.

$$\begin{aligned} sw^h(s_1, d_2) &= 9 \\ sw^c(s_1, d_2) &= 10 \end{aligned}$$

b. It₂ viciously attacked the₁ black chihuahua.

$$\begin{aligned} sw^h(s_2, d_1) &= 10, & sw^h(s_2, d_2) &= 9 \\ sw^c(s_2, d_1) &= 9, & sw^c(s_2, d_2) &= 10 \end{aligned}$$

c. { ?? The₁ dog (H) / The₁ black dog (C) } barked loudly.

$$\begin{aligned} sw^h(s_3, d_1) &= 9, & sw^h(s_3, d_2) &= 8 \\ sw^c(s_3, d_1) &= 10, & sw^c(s_3, d_2) &= 0 \end{aligned}$$

The example is first discussed from the Praguian point of view, using sw^h as defined above. In (10)a the modified algorithm produces the description *the white chihuahua* to refer to d_2 . Thereby, d_2 becomes the most salient entity, with the highest-but-one degree of salience, as it is introduced in the topic of the sentence. In (10)b a new entity, d_1 , is introduced to the discourse using the description *the black chihuahua*. Since this is done in the focus of the sentence, d_1 rises to maximal salience. Entity d_2 is referred to by a pronoun occurring in the topic of the sentence, so its salience weight is unchanged.¹² In (10)c, d_1 is referred to for the second time, and as it is currently the most salient dog, the modified algorithm generates the description *the dog*. This result, though marginally acceptable, is not optimal. A simple solution to this problem is to ignore small differences in activation degree between competitors (see also Kruijff-Korbayová and Hajičová 1997:41). This would amount to re-defining $\text{Dist}(r, s)$ in such a way that the distractor

¹²The problem of pronoun generation is addressed in Section 4.5.1 below.

set also includes entities that have a salience weight in s which is, say, one point lower than the salience weight in s of r itself. Reiter (p.c.) has suggested that such a salience threshold might be dependent on text genre. For instance, the salience threshold for legal texts seems to be very high (see e.g., Maes 1991, who shows that legal texts hardly contain anaphoric descriptions).

Let us now discuss example (10) from the Centering perspective. After the first sentence has been uttered, C_f ((10)a) is the singleton set containing d_2 , and as a result this entity is now the most salient entity. In (10)b both d_2 and d_1 are referred to, and C_f ((10)b) therefore contains both d_1 and d_2 . Since d_2 is referred to in subject position, it is ranked higher than d_1 . Consequently, to refer to d_1 in (10)c the modified Incremental Algorithm correctly produces *the black dog*.

Now consider the following discourse, based on the dog show example from Section 4.3.1. Suppose Joe went to the dog show and bought the dogs d_{45} and d_{53} .

- (11) a. Joe bought the₄₅ large black long-haired sausage dog and the₅₃ small grey pygmy poodle with the perm wave.

$$\begin{aligned} sw^h(s_1, joe) &= 9, & sw^h(s_1, d_{45}) &= 10, & sw^h(s_1, d_{53}) &= 10 \\ sw^c(s_1, joe) &= 10, & sw^c(s_1, d_{45}) &= 9, & sw^c(s_1, d_{53}) &= 9 \end{aligned}$$

- b. The₄₅ sausage dog was a bargain.

$$\begin{aligned} sw^h(s_2, joe) &= 8, & sw^h(s_2, d_{45}) &= 9, & sw^h(s_2, d_{53}) &= 8 \\ sw^c(s_2, joe) &= 0, & sw^c(s_2, d_{45}) &= 10, & sw^c(s_2, d_{53}) &= 0 \end{aligned}$$

- c. { The₅₃ poodle (H) / ?? The₅₃ small grey pygmy poodle with the perm wave (C) } was very expensive though.

$$\begin{aligned} sw^h(s_3, joe) &= 7, & sw^h(s_3, d_{45}) &= 8, & sw^h(s_3, d_{53}) &= 9 \\ sw^c(s_3, joe) &= 0, & sw^c(s_3, d_{45}) &= 0, & sw^c(s_3, d_{53}) &= 10 \end{aligned}$$

Comparing the two different salience weight assignments in this example presents the following picture. After generation of the first sentence, both approaches assign high salience weights to *joe* and the dogs d_{45} and d_{53} . The second sentence only contains a reference to d_{45} (*the sausage dog*). Using the Centering approach, this reduces the salience weight of *joe* and d_{53} to zero as they are not mentioned in (11)b, whereas using the Hajičová approach entails that the salience weights of *joe* and d_{53} are reduced much less. This difference in salience weight reduction for d_{53} greatly influences its description in (11)c. Seen from a Praguian

perspective, d_{53} is still the most salient poodle at this stage, and the generated description is *the poodle*. However, from the Centering perspective, d_{53} is not salient at all and the modified Incremental Algorithm again produces the description *the small grey pygmy poodle with the perm wave*, just as it did for the first-mention of this dog in (11)a. This shows that the Centering assumption that only entities mentioned in the previous sentence can have a non-zero salience weight, is too strong from the perspective of definite descriptions. Again, there is an obvious way to remedy this shortcoming: the structure of the discourse should be taken into account (see e.g., Walker 1998 for a proposal to this effect).

4.4.4 Discussion: revising salience weight assignment

There are several principled ways to determine salience weights that can be used in the modified algorithm. Here, two of them have been discussed and compared: one is based on the hierarchical focusing constraints of Hajičová 1993 and the other on the Centering approach of Grosz et al. 1995. Probably the most important distinctions between the two are (i) the Praguian assumption that information introduced in the focus of an utterance has a somewhat higher salience weight than information introduced in the topic, as opposed to the Centering assumption that information referred to in subject position is more salient than information referred to in object position,¹³ and (ii) the Centering assumption that only entities mentioned in the previous sentence can have a non-zero salience weight, while in the Praguian approach the determination of salience weights is not restricted to the previous sentence.

Two examples have been used to illustrate the differences between the two approaches. In the first case, the Centering approach yields better results, while in the second case, it is the Hajičová way of determining salience weights which pays off. These examples suggest that the Centering ordering of salience is preferable over the Praguian topic/focus ordering, but that a gradual decrease in salience of non-mentioned entities gives better results than an abrupt loss of salience. An obvious step is therefore to combine the Centering definition of salience with a mechanism for gradual decrease. Here, the most simple form of decrease is assumed, where the salience of all discourse entities that have not been mentioned in the current sentence decreases by one. (This simplification of the Praguian approach is sufficient for current purposes.) This leads to Definition 5:

¹³In most cases, but not always, the subject of a sentence refers to topical information. In those cases, the two approaches make different predictions.

Definition 5 (Revised salience weight assignment)

Let U_i be a sentence uttered in the context of sw_i , in which reference is made to $\{d_1, \dots, d_n\} \subseteq D$. Let $C_f(U_i)$ (the forward looking centers of U_i) be a partial order defined over $\{d_1, \dots, d_n\} \subseteq D$. Then the new salience function sw_{i+1} is defined as:

$$sw(s_0, d) = 0 \text{ for all } d \in D$$

$$sw(s_{i+1}, d) = \begin{cases} \text{level}(d, C_f(U_i)) & \text{if } d \in C_f(U_i) \\ \max(0, sw_i(d) - 1) & \text{if } d \notin C_f(U_i) \text{ and } d \in C_f(U_j), j < i \end{cases}$$

Using the above definition, the modified algorithm produces the intuitively correct results for the examples discussed so far (as the reader may check). Although Definition 5 may still have to be refined to account for salience fluctuations in more complicated examples, e.g., containing complex grammatical constructions, it provides us with a workable salience function to be used in the modified algorithm.

For the generation of reduced anaphoric descriptions, it is very important to have some form of salience weight assignment, ensuring that entities which have been recently mentioned are regarded as (much) more salient than the other entities in the domain. Without such a mechanism, the `MakeReferringExpression` algorithm would generate a full description of an entity at every mention of this entity, using the same description throughout the output text. For the generation of reduced definite descriptions, in most cases it seems sufficient to only keep track of large differences in salience weight (e.g., between mentioned versus unmentioned entities); small differences (e.g., between entities that have been mentioned in different positions in the same sentence) appear to be mainly relevant for pronominalisation (discussed in Section 4.5.1). So far, the ‘salience threshold’ has been set to one: a referent r counts as being more salient than other entities in the domain if its salience weight is at least one point higher than that of those other entities (in other words, only those entities that are at least equally salient as r count as its distractors). However, to stay on the safe side the threshold might be set higher. A consequence of such a measure would be the generation of fewer reduced descriptions, which reduces the risk of ambiguity but also reduces coherence of the generated texts. In general, the choice between fewer or more reduced anaphoric descriptions involves a trade-off between incoherence and ambiguity. Less ambiguity (due to fewer reduced descriptions) entails more incoherence, and vice versa. The desired ratio between the two depends among other things on text

genre; for instance, in legal texts coherence is far less important than unambiguous reference, as observed by Maes 1991.

Finally, a word on the computational consequences of adding a salience function to the algorithm. In general, the determination of salience adds little computational overhead. To compute a new salience function only the values of the entities mentioned in the current clause and the entities with a non-zero salience weight have to be updated. Even for huge domains, the latter set is highly restricted, containing only the entities mentioned in the last few sentences.

4.5 Further extensions

So far, it has been shown in some detail how the modified Incremental Algorithm can generate reduced definite descriptions which are only distinguishing in context. In addition, it seems that the modified algorithm can very well serve as a basis for the generation of other kinds of referring expressions, such as pronouns, relational and bridging descriptions. In this section, three further extensions of the (modified) Incremental Algorithm are sketched. Section 4.5.1 discusses a simple form of pronominalisation within the modified algorithm, Section 4.5.2 shows how relational descriptions can be generated by slightly modifying the algorithm and Section 4.5.3 outlines how the combination of the two preceding extensions paves the way for the generation of bridging descriptions.

4.5.1 Pronominalisation

Reiter and Dale (1997:81) point out that a simple but surprisingly effective strategy for pronominalisation is to “use a pronoun to refer to an entity if the entity was mentioned in the previous clause, and there is no other entity in the previous clause that the pronoun could possibly refer to”. This is a fairly conservative strategy, which has the advantage that it will not often produce incorrect pronominalisations. On the other hand, it has been claimed that a pronoun should be used whenever this is possible.¹⁴ The presence of salience weights in the modified algorithm makes it possible to employ a somewhat less conservative strategy, taking the pronominalisation decision *within* the algorithm. The basic idea is as follows: if an entity r is the single most salient entity in the domain (and therefore has no distractors), and there is an antecedent for this entity in the direct linguistic context, then r can be referred to by using a pronoun. The required modifications to the algorithm are shown in Figure 4.5.

¹⁴This is a specific instance of the DOAP principle from Williams 1997: “Don’t Overlook Anaphoric Possibilities.”

```

MakeReferringExpression ( $r, P, s$ )
 $P \leftarrow \text{PreferredAttributes}(\text{Datatype}(r))$ ,
 $RemDist \leftarrow \text{Dist}(r, s)$ ,
 $tree \leftarrow \text{nil}$ 
if  $RemDist = \{\}$   $\wedge \exists c : \text{Antecedent}(c, r)$ 
then  $tree \leftarrow \text{Pronominalise}(tree, r)$ 
       return  $tree$ 
else for each member  $A_i$  of list  $P$  do
        $\vdots$ 

```

FIGURE 4.5: Pronominalisation within MakeReferringExpression. The remainder of the algorithm is as given in Figure 4.2.

This is certainly not offered as the final answer to the problem of generating pronouns. One obvious limitation is that it does not take the role of semantics and common sense into account (see e.g., Kameyama 1996, Passonneau 1996). The current approach is only concerned with the default approach to pronoun generation, which is purely syntactically motivated and does not address how semantic or pragmatic information can override the default. The success of this strategy depends fully on the adequacy of the underlying model of salience weight determination, and here Definition 5, presented in Section 4.4.4, could serve as a good starting point. The following example may serve as an illustration. If sentence (12)a has been uttered at the onset of a discourse, then according to Definition 5 entity d_2 is more salient than entity d_3 . If the modified algorithm subsequently generates a referring expression for d_2 , it will produce *it*, while a subsequent reference to d_3 will yield a full, anaphoric description. This is intuitively right in the sense that most people would interpret *it* as referring to the white chihuahua.¹⁵

- (12) a. The₂ white chihuahua was chasing the₃ cat.
 b. {It₂/The₃ cat} ran fast.

In pronominalisation, the notion of a ‘salience threshold’ and the trade-off between incoherence and ambiguity are particularly relevant, because pronouns are even more likely to give rise to ambiguities than reduced definite descriptions. For instance, the pronoun *it* in (12) is ambiguous in the sense that it is most likely to refer to d_2 , but it could also be interpreted as referring to d_3 . This ambiguity can be avoided by rais-

¹⁵This intuition corresponds to the Centering Theory claim that CONTINUE transitions are preferred over RETAIN transitions (Grosz et al. 1995; see also Section 3.5.6). This claim has been experimentally confirmed by Gordon et al. 1993, Walker et al. 1994 (for Japanese), Hudson-D’Zmura and Tanenhaus 1998, and others.

ing the salience threshold so that d_2 no longer counts as the single most salient entity in the domain. Given the increased salience threshold, d_3 now counts as a distractor from which d_2 has to be distinguished, and this results in the non-ambiguous description of d_2 as *the dog*.

4.5.2 Relational descriptions

The preceding sections have only addressed the description of entities in terms of simple attributes such as colour, size and type. Here, the proposed approach is extended to include *relational descriptions*, which describe an intended referent a in terms of its relationship to another entity b . An example is *the dog with the bone*. Following Levelt 1989, b (here, the bone) is referred to as the *relatum* of a .

Dale and Haddock 1991 offer an algorithm for the generation of relational descriptions based on the “Greedy Heuristics algorithm” (Dale 1992). The basic claim made here is that it is possible to generate relational descriptions using a slightly adapted version of the modified Incremental Algorithm and that this has some interesting consequences.¹⁶

Extending the domain

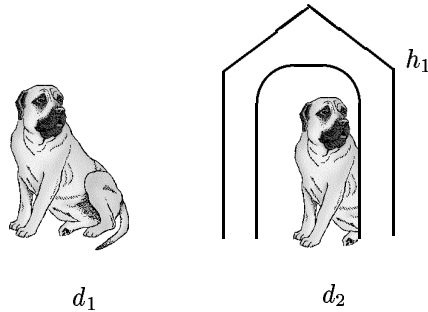
To be able to generate relational descriptions, the domain database must be extended with *relations*: data structures providing information about the relationships between the entities in the database. A relation \mathcal{R} is a data structure of data type *relation*. This general data type has several subtypes, e.g., *spatial* for spatial relations and *possessive* for possessive relations. (In this section, only spatial relations are discussed.) Each data structure of type *relation* has three attributes: (i) *type*, which specifies the type of the relation, e.g., *in* or *left_of*, (ii) *arg_1*, which specifies the first argument of the relation by its index, and (iii) *arg_2*, which specifies the second argument. (For the sake of simplicity it is assumed that all relations have two arguments, leaving aside relations that have more arguments, such as *between*.) Relations are abbreviated as $[R, R\text{-type}, [A1, A2]]$, where R is the data type of the relation (e.g., *spatial*), $R\text{-type}$ is the value of its *type* attribute (e.g., *in*), and $[A1, A2]$ is the list of its arguments *arg_1* and *arg_2* respectively.

The inclusion of relations in the database can be illustrated using sample domain D_2 , given in the form of data structures in Figure 4.6 and pictured below it. Domain D_2 contains two data structures of type *spatial*: one saying that d_2 is inside h_1 and one saying that d_1 is located to the left of h_1 . Of course, d_1 is also located to the left of d_2 . However, to keep matters relatively simple, the database is assumed to contain

¹⁶A variant of the algorithm proposed in this section has been implemented by Sebastiaan van Erk and André Verleg.

only spatial relations between entities that are physically close to each other.¹⁷ Thus, the database representation of D_2 does not contain a relation between entities d_1 and d_2 , because both d_1 and d_2 have a closer relation with the doghouse h_1 than with the other dog.

<i>animal</i> referent : d_1 type : mastiff size : large colour : grey		<i>spatial</i> type : left_of arg_1 : d_1 arg_2 : h_1
<i>animal</i> referent : d_2 type : mastiff size : large colour : grey		<i>spatial</i> type : in arg_1 : d_2 arg_2 : h_1
<i>object</i> referent : h_1 type : doghouse size : large colour : white		

FIGURE 4.6: Domain D_2 .

A subsumption hierarchy is assumed to exist on the type value of certain relations. Figure 4.7 shows a relevant portion of such a subsumption hierarchy for spatial relations.¹⁸ The values in and next_to are assumed

¹⁷The problem of deciding which entities are ‘close’ may be likened to the problem of deciding which entities are ‘large’. In a realistic database, both size and spatial distances would be given in absolute measures (cm/in/pt) and a reasoning component would have to determine which entities are indeed large or nearby. For now, the input data are simply assumed to be given.

¹⁸It is interesting to note that most of the work on categorisation in cognitive science

to be basic level values; the subsuming σ can be thought of as the under-specified spatial relation (“there is some spatial relation between these objects”). The type value stored for relations in the domain is the most specific one, just as for the animal types. The advantages of having subsumption hierarchies for relation-types are discussed below.

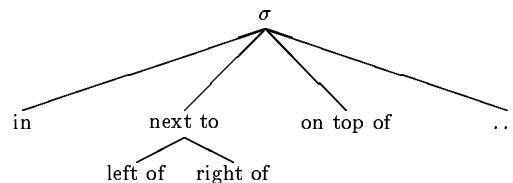


FIGURE 4.7: Part of the subsumption hierarchy on spatial relations

Outline

Before it can be shown how the `MakeReferringExpression` algorithm can be extended to include relations in the description of objects, some reflection is needed on the ordering of relations versus properties, and the ordering of relations among themselves. It seems an acceptable assumption that people prefer to describe an entity in terms of simple properties, and only shift to relations when properties do not suffice (i.e., properties stand to relations as absolute properties stand to relative properties). This follows from the omnipresent *principle of least effort* (e.g., Zipf 1949, Clark and Wilkes-Gibbs 1986): it takes less effort to consider and describe only one entity. How the different kinds of relations (spatial, possessive, etc.) should be ordered is somewhat less obvious; however, it seems likely that people have a preference for relations that are easily perceivable, as is the case with properties. From this it follows that spatial and part-of relations will be preferred over relations that are (usually) less easy to perceive, such as possessive relations. Reasonable as such assumptions seem, they are not based on psycholinguistic research into these issues. In the current section, therefore, it is simply assumed that properties are preferred over relations and that spatial relations are preferred over other kinds of relations.

Figure 4.8 shows a first version of the modified Incremental Al-

has been concerned with physical entities, and not with relations. A notable exception is Case study 2 in Lakoff 1987. Here Lakoff studies the interrelations between various senses of the preposition *over*, which leads – on a lower level – to a similar structure as shown in Figure 4.7. However, as it appears that the notion of basic level values for certain classes of relations has not been studied, no psychological reality is claimed for the assumptions made here.

```

MakeReferringExpression ( $r, s$ )
 $P \leftarrow$  PreferredAttributes (Datatype ( $r$ ))
 $PR \leftarrow$  PreferredRelations (Datatype ( $r$ )) *
 $RemDist \leftarrow$  Dist( $r, s$ )
 $tree \leftarrow$  nil
for each member  $A_i$  of list  $P$  do % properties
   $V \leftarrow$  FindBestValue ( $r, A_i, \text{BasicLevelValue} (r, A_i), RemDist$ )
   $OutRuledRefs \leftarrow$  RulesOut ( $RemDist, \langle A_i, V \rangle$ )
   $tree' \leftarrow$  UpdateTree ( $tree, \langle A_i, V \rangle$ )
  if ( $OutRuledRefs \neq \{\}$   $\vee A_i = \text{type}$ )  $\wedge tree' \neq \text{nil}$ 
  then  $tree \leftarrow tree'$ 
     $RemDist \leftarrow RemDist - OutRuledRefs$ 
  endif
  if  $RemDist = \{\}$ 
  then  $tree \leftarrow$  AddDefDet ( $tree$ )
    return  $tree$ 
  endif
endfor
for each member  $R_i$  of list  $PR$  do % relations
  for each relation of the form  $[R_i, R\text{-type}, [r, r']]$  do *
     $VR \leftarrow$  FindBestValue_Rel ( $r, [R_i, R\text{-type}, [r, r']],$ 
      BasicLevelValue_Rel ( $R\text{-type}$ ),  $RemDist$ )
     $OutRuledRefs \leftarrow$  RulesOut_Rel ( $RemDist, [R_i, VR, [r, r']]$ )
     $tree' \leftarrow$  UpdateTree_Rel ( $tree, [R_i, VR, [r, r']]$ )
    if  $OutRuledRefs \neq \{\}$   $\wedge tree' \neq \text{nil}$ 
    then  $tree \leftarrow tree'$ 
       $RemDist \leftarrow RemDist - OutRuledRefs$ 
       $tree2 \leftarrow$  MakeReferringExpression ( $r', s$ ) *
       $tree \leftarrow$  AddTree ( $tree, tree2$ ) *
    endif
    if  $RemDist = \{\}$ 
    then  $tree \leftarrow$  AddDefDet ( $tree$ )
      return  $tree$ 
    endif
  endfor
endfor
 $tree \leftarrow$  AddIndefDet ( $tree$ )
return  $tree$ 

```

FIGURE 4.8: Extension of the modified Incremental Algorithm which incorporates relational descriptions (version 1). A * marks the main differences in the treatment of relations as opposed to properties.

gorithm which allows for the generation of relational descriptions through *recursion*. (Below, this version is replaced by a more sophisticated version.) The main additions to the algorithm are the following.

First of all, a new variable *PR* is added, which contains relation data types such as *spatial*, *part-of*, *possessive*, etc. in order of preference. Like the list of preferred attributes *P*, *PR* is initialised relative to the data type of referent *r*. Second, a new for loop dealing with relations is added to the algorithm. This loop is reached if list *P* of preferred attributes is exhausted and no distinguishing description has been produced. The selection of a relation \mathcal{R} to be included in a description proceeds in almost exactly the same way as the selection of a property. The functions used in the selection of relations, which are discussed in more detail below, closely correspond to the functions used in the selection of properties. The main difference with the selection of properties is that when a relation is included, a recursive call of the algorithm is made to describe the relatum r' of *r*. After this recursive call of the algorithm has finished, the syntactic tree for the relatum (stored in *tree2*) is incorporated in the tree that has so far been constructed for the original referent.

Let us walk through the new algorithm. Its first part has not changed much, except for the addition of the variable *PR*. After initialisation of the variables *P*, *PR*, *RemDist* and *tree*, the algorithm first iterates through *P*. If *P* is exhausted before the set of remaining distractors is empty, the algorithm continues by iterating through the list of preferred relation data types *PR*. Because a domain entity may be involved in more than one relation of the same data type (for instance, it may be located to the left of one entity, and to the right of another), an embedded loop is entered for each data type: the algorithm goes through all relations in the database which are of that data type and have *r* as their first argument.¹⁹ For each of these relations, first the function *FindBestValue_Rel* is called to determine the best value of this relation's type attribute. This function is shown in Figure 4.9.

FindBestValue_Rel works in much the same way as *FindBestValue* works for properties. The main difference is that it is necessary to keep track of the relation being currently dealt with, since a referent *r* may be involved in more than one relation. Therefore the current relation \mathcal{R} is passed on in its entirety as the second argument of *FindBestValue_Rel*. The other arguments of the function are the same as in the version for attributes. So, *FindBestValue_Rel* takes as input the referent *r*, the relation \mathcal{R} under consideration, the basic level value of this relation's type

¹⁹The current algorithm only considers relations with *r* as their first argument. This simplifying approach will be abandoned later.

```

FindBestValue_Rel ( $r, [R, R\text{-type}, [r, r']], \text{initial-value}, \text{RemDist}$ )
if UserKnows_Rel ( $r, [R, \text{initial-value}, [r, r']]$ ) = true
then  $value \leftarrow \text{initial-value}$ 
else  $value \leftarrow \text{novalue}$ 
endif
if ( $msv \leftarrow \text{MoreSpecificValue\_Rel}(r, R\text{-type}, value) \neq \text{novalue} \wedge$ 
  ( $nv \leftarrow \text{FindBestValue\_Rel}(r, [R, R\text{-type}, [r, r']], msv, \text{RemDist}) \neq \text{novalue} \wedge$ 
  | RulesOut_Rel ( $\text{RemDist}, [R, nv, [r, r']]$ ) | >
  | RulesOut_Rel ( $\text{RemDist}, [R, value, [r, r']]$ ) |
then  $value \leftarrow nv$ 
endif
return  $value$ 

RulesOut_Rel ( $\text{RemDist}, [R, VR, [r, r']]$ )
if  $VR = \text{novalue}$ 
then return {}
else return { $x : x \in \text{RemDist} \wedge$ 
   $\neg \exists y \in D : \text{UserKnows}(x, [R, VR, [x, y]]) = \text{true}$  }
endif

```

FIGURE 4.9: Functions used in Figure 4.8.

attribute, and the set of remaining distractors *RemDist*. The relation's type value is stored in the variable *R-type*. Its basic level value is retrieved by the function *BasicLevelValue_Rel*. The only argument of this function is *R-type*. Giving referent *r* and the current data type R_i as input (which would correspond most closely to *BasicLevelValue*) does not provide enough information, because *r* may be involved in different relations of data type R_i . Given the presence of hierarchies such as the one in Figure 4.7, knowing the most specific type value of a relation is sufficient to determine its basic level value.

Other functions used in determining the best value are *UserKnows_Rel* and *MoreSpecificValue_Rel*. Like *FindBestValue_Rel*, these functions are similar to their non-relational counterparts, except that *MoreSpecificValue_Rel* takes *R-type* (and not *R*) as its second argument. Again, this is because *r* may be involved in more than one relation of the same data type, and it is essential to know which one is meant. The function *RulesOut_Rel* is somewhat more complicated than its non-relational counterpart, and is therefore shown in Figure 4.9. Starting (for the time being) from the assumption that the relatum is the second argument of the relation, it selects those elements from *RemDist* of which the user

knows that they do not have a relation of data type R and relation-type VR with another domain object as the second argument.

After the current relation's best type value has been determined, *OutRuledRefs* and *tree'* are instantiated, using the functions *RulesOut_Rel* (discussed above) and *UpdateTree_Rel* respectively. The main difference between *UpdateTree_Rel* and *UpdateTree* for attributes is that the former adds an *incomplete* expression to the tree: an NP slot is left open where an expression for the relatum can be filled in. Most relations are expressed by a PP, with as its head an expression for VR . The NP which is part of the PP is left open, to be filled in later with an expression for the relatum. Some relations may be expressed by a relative clause. In general, relations should be expressed in such a way that they do not give rise to ambiguities, e.g., due to scoping problems (see Horacek 1997). If this is not possible, *UpdateTree_Rel* returns a nil tree.

After instantiation of *OutRuledRefs* and *tree'* the algorithm checks if *OutRuledRefs* is not the empty set and if *tree'* is not nil. If both conditions are met, *tree* is replaced by *tree'* and the set of *OutRuledRefs* is removed from the set of remaining distractors. Now the main difference with the treatment of attributes is encountered: *MakeReferringExpression* is called for the relatum r' , and the resulting description is incorporated in the tree for r , using the function *AddTree*. After this, the algorithm checks if the set of remaining distractors is empty. If so, a definite determiner is added to the tree and the resulting description is returned. Otherwise, the algorithm considers the next available relation. If the algorithm has inspected all available relations without producing a distinguishing description, an indefinite description is returned.

Examples

In order to illustrate the generation of relative descriptions using the algorithm of Figure 4.8, let us discuss an example. Suppose that entity d_2 of example domain D_2 must be described in the initial situation s_0 (all entities are equally non-salient). After the function *MakeReferringExpression* (d_2, s_0) has been called, the list of preferred properties P is instantiated as $\langle \text{type, colour, size} \rangle$. The list of preferred relation data types PR is assumed to be $\langle \text{spatial} \rangle$. *RemDist* is initialised as the set of distractors of d_2 in s_0 , i.e., $\{d_1, h_1\}$.

The algorithm first iterates through P , and the first property it encounters is $\langle \text{type, mastiff} \rangle$. As both dogs in D_2 are mastiffs, the best value for the type attribute of d_2 is *dog*, which rules out the doghouse h_1 . This property is realised as the N^0 in the NP tree under construction, and h_1 is removed from the set of remaining distractors. This does not leave the set empty: the second dog d_1 is not yet ruled out. The

second and third attributes from P (colour and size) fail to distinguish d_2 from d_1 , and are not included. List P is exhausted now, but the set of remaining distractors is not yet empty and no definite description has been returned. Therefore, the algorithm starts to inspect the list of preferred relation datatypes PR . The first datatype on the list is *spatial*. Referent d_2 happens to be involved as the first argument in only one spatial relation, $\langle \textit{spatial}, \textit{in}, [d_2, h_1] \rangle$. This relation is retrieved from the database and `FindBestValue_Rel` is called to determine the best value of its type attribute. For relation-type *in*, the basic level value is *in* itself, so out of necessity this is also the best value, and VR is instantiated as *in*. Next, $OutRuledRefs$ is instantiated as the set of those remaining distractors that are ruled out by $\langle \textit{spatial}, \textit{in}, [d_2, h_1] \rangle$. This is the set $\{d_1\}$ containing the only remaining distractor, which is not inside anything. Then the variable $tree'$ is instantiated as the result of `UpdateTree_Rel` ($tree, \langle \textit{spatial}, \textit{in}, [d_2, h_1] \rangle$). This tree is shown as tree I in Figure 4.10. After checking that $OutRuledRefs$ is not empty and $tree'$ is not nil, $tree$ is instantiated as $tree'$ and the set $OutRuledRefs$ is subtracted from the set of remaining distractors (which leaves the latter set empty).

Now the algorithm enters the recursion: the function `MakeReferring-Expression` is called with as parameters h_1 (the relatum) and the state s_0 . First P and PR are initialised with respect to the datatype of h_1 , which is *object*. For this datatype, like for *animal*, P is assumed to be $\langle \textit{type}, \textit{colour}, \textit{size} \rangle$ and $PR = \langle \textit{spatial} \rangle$. Then $RemDist$ is initialised as the set of distractors of h_1 . This is the set containing both other domain entities $\{d_1, d_2\}$ (all entities in the domain still have the same salience). The algorithm then examines the first element on the list of

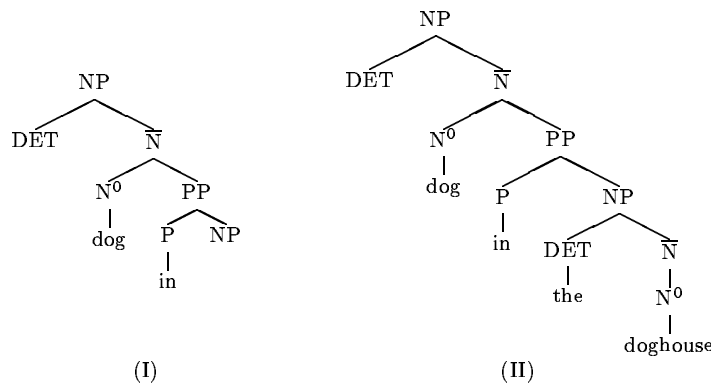


FIGURE 4.10: Two crucial stages in the generation of *the dog in the doghouse*.

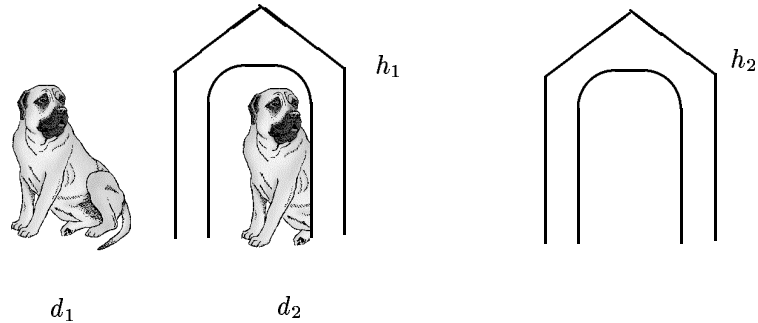
preferred attributes, i.e., the type attribute. The type of h_1 is doghouse. This property rules out the dogs d_1 and d_2 , so an expression for it is added to the tree for h_1 . Subtracting d_1 and d_2 from *RemDist* leaves the set empty: h_1 is the most salient doghouse in the domain. This means that the selection of properties has finished successfully, so a definite determiner is added to the tree under construction and the recursive call of the algorithm returns a syntactic tree for *the doghouse*, which is stored as *tree2*. Then the initial call of *MakeReferringExpression* continues, and the function *AddTree* inserts *tree2* into the empty NP slot of the original tree for d_2 . The result is shown as tree II in Figure 4.10. At this point, the set of remaining distractors of d_2 is empty: d_2 is the most salient dog inside another entity. To wrap things up *AddDefDet* inserts a definite article into the main NP and the final tree is returned. Thus, the algorithm generates *the dog in the doghouse*, which is a distinguishing, relational description of d_2 .

In order to show why it is useful to determine the best value for a relation's type attribute, let us briefly discuss another example: the generation of a description for entity d_1 in domain D_2 , again in state s_0 . For the first part, the algorithm proceeds as in the previous example. The property $\langle \text{type, dog} \rangle$ is selected and an expression for it is incorporated in the tree for d_1 . Again, colour and size do not help to distinguish d_1 from d_2 , so the algorithm goes on to check the relations in which d_1 is involved. Like d_2 , d_1 is only involved in a single relation: $\langle \text{spatial, left_of, } [d_1, h_1] \rangle$. When trying to find the best value for this relation's type attribute, it turns out that this is the basic level value *next_to*. The more specific value *left_of*, which is stored in the database, is not the best value since it fails to rule out more distractors than *next_to*: both rule out the only remaining distractor d_2 . Since *next_to* and *left_of* both rule out the same number of distractors, the more general value of the two is selected, which is *next_to*. The rest of the generation procedure is similar to the previous example, resulting in the description of d_1 as *the dog next to the doghouse*. Simply expressing the type value stored in the database, instead of searching for the best value, would have resulted in the description *the dog left of the doghouse*. This description is overly specific, since the information that d_1 is located to the *left* of doghouse h_1 is irrelevant in the current domain. However, as the reader may check, the addition to the domain of a dog which is located to the right of h_1 would lead to the description of d_2 as *the dog left of the doghouse*.

Revision

It has been shown how, with only a few modifications to the algorithm, relational descriptions can be generated. The modifications can be sum-

marised as follows: (i) in the case that all attributes have been checked and no distinguishing description has been generated, the algorithm tries to include relations in the description, in a way that is similar to the selection of properties; (ii) when a relation is included, a recursive call of the algorithm is made to construct a tree for the relatum, which is incorporated in the tree under construction. This approach works well for domains like D_2 , where the relatum included in a description can be uniquely identified using unary properties only. However, when the relatum can be uniquely identified only by means of its relation to the original referent, the algorithm as shown in Figure 4.8 does not produce the desired result. This can be illustrated by extending domain D_2 with a second white doghouse, h_2 , which is assumed (for the sake of argument) to have no spatial or other relation to any of the other entities in the domain (it is not ‘close’ to anything). The resulting domain is D_3 , shown below.



Now assume that a description must be generated for d_2 in domain D_3 , using the algorithm in Figure 4.8. At first, the algorithm proceeds as before: first, the type property of d_2 is included (ruling out h_1 and h_2), and then the spatial relation with h_1 is included (ruling out d_1). However, when coming to the point where h_1 , the relatum of d_2 , must be described using a recursive call of the algorithm, it becomes apparent that including its type property only rules out the dogs d_1 and d_2 , but not h_2 , the other doghouse. Since h_1 is not involved as the first argument in any spatial relation, no relations are found that can help to distinguish h_1 from h_2 . The algorithm therefore returns the indefinite description *a doghouse* for h_1 , resulting in a description of d_2 as *the dog in a doghouse*. Although this is a correct distinguishing description of d_2 , it would be far more natural to describe d_2 as *the dog in the doghouse*, using a definite description to refer to doghouse h_1 . The current version of the algorithm cannot generate such a description, since the

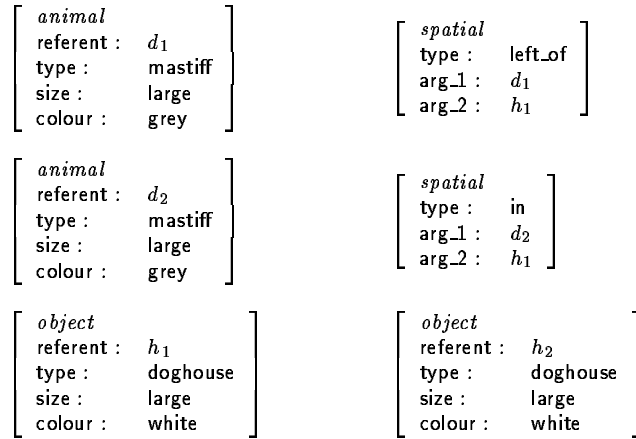


FIGURE 4.11: Domain D_3 .

only thing that distinguishes h_1 from h_2 , i.e., the relation $\langle \textit{spatial}, \textit{in}, [d_2, h_1] \rangle$, is not taken into account. This can be remedied by having the algorithm consider *all* relations in which a referent r is involved, regardless whether r is the first or the second argument. Let us assume that `MakeReferringExpression` has been changed in this way. (Below, the required modifications to the algorithm are discussed.) Now, when generating a description for h_1 , the relation $\langle \textit{spatial}, \textit{in}, [d_2, h_1] \rangle$ can be considered. Since this relation rules out the single remaining distractor h_2 , the algorithm tries to include it in the description of h_1 . However, a serious problem is now encountered: including the relation between d_2 and h_1 in the description of h_1 leads to an endless recursion, since it causes `MakeReferringExpression` to be called again for a description of d_2 as the relatum of h_1 . In describing d_2 , the algorithm will use its relation with h_1 again, and so on *ad infinitum*. In short, the algorithm will get into an endless cycle of recursion, as it will try to include the same relation over and over in the description of both referent and relatum. This is illustrated in (13) below.

- (13) The dog in the doghouse containing the dog in the doghouse containing ...

In order to avoid this infinite recursion and still obtain the description *the dog in the doghouse* for d_2 , it is necessary to make use of the knowledge that the relation between d_2 and h_1 is already included in the description of d_2 , and therefore does not have to be included in the description of h_1 anymore. In order to do this, the `MakeReferringExpression`

algorithm needs to be further adapted as shown in Figure 4.12. In this new version, a new parameter is added to pass on the relation \mathcal{R} in a recursive call of the algorithm. This relation already contains information about the relatum, and can therefore be used to rule out some of its distractors in advance: in the new version of the algorithm, the set of remaining distractors *RemDist* is initialised as the set of distractors of r in s minus those distractors that are already ruled out by \mathcal{R} . In the initial call of the algorithm, \mathcal{R} is instantiated as ‘the empty relation’ $\langle \textit{relation}, \textit{novalue}, [\textit{nil}, \textit{nil}] \rangle$, which does not rule out any distractors.

Figure 4.12 also shows the modifications that are required if all relations in which r is involved as either the first or the second argument are to be considered. In the new algorithm, variables are used for the arguments of a relation (notation: $[R, R\text{-type}, [A1, A2]]$) and r is given as an extra parameter to *RulesOut_Rel*; this is necessary to determine if r is the first or second argument of the relation, so as to rule out the correct distractors. If r is the first argument of the relation ($r = A1$) then *RulesOut_Rel* operates as before. If r is the second argument, all remaining distractors are ruled out that are involved in such a relation with another entity as the first argument. The new version of *RulesOut_Rel* is given in Figure 4.13. In addition, *UpdateTree_Rel* also needs to have r as an extra argument in order to determine if r is the first or the second argument of the relation. This is relevant for the wording chosen to express the relation. For instance, if r is the first argument of a relation with type *in*, then the relation will be expressed as *in X* (where X represents the empty slot for the description of r'); on the other hand, if r is the second argument of the relation the chosen expression will be *containing X* or *with X in it*.

To illustrate the working of the new version of the algorithm, assume that a description for d_2 in domain D_3 must be generated. The algorithm is then called with parameters d_2 , the initial state s_0 and the new parameter \mathcal{R} , which is initialised as $\langle \textit{relation}, \textit{novalue}, [\textit{nil}, \textit{nil}] \rangle$. The lists P and PR are instantiated as before. *RemDist* is the set of all distractors of d_2 , i.e., $\{d_1, h_1, h_2\}$; no distractors are ruled out by \mathcal{R} . Like before, the algorithm first adds an expression for the type property of d_2 to the tree, ruling out the doghouses h_1 and h_2 . Then the *in* relation with h_1 is incorporated to rule out the other dog, d_1 , which is not inside anything. Now, a recursive call of the algorithm is made to construct a tree for h_1 , the relatum of d_2 . The parameters are h_1 , state s_0 and, crucially, the relation between d_2 and h_1 : $\langle \textit{spatial}, \textit{in}, [d_2, h_1] \rangle$.

The fact that this time, \mathcal{R} is not ‘empty’ has consequences for the initialisation of the set of remaining distractors of h_1 : since none of the distractors of h_1 contain another domain entity, \mathcal{R} rules out all of them

```

MakeReferringExpression ( $r, s, \mathcal{R}$ ) *
 $P \leftarrow$  PreferredAttributes (Datatype ( $r$ ))
 $PR \leftarrow$  PreferredRelations (Datatype ( $r$ ))
 $RemDist \leftarrow$  Dist( $r, s$ ) - RulesOut_Rel (Dist ( $r, s$ ),  $\mathcal{R}, r$ ) *
 $tree \leftarrow$  nil
for each member  $A_i$  of list  $P$  do
:
endfor
for each member  $R_i$  of list  $PR$  do
  for each relation of the form [ $R_i, R\text{-type}, [A1, A2]$ ]
  where  $r = A1$  or  $r = A2$  do *
     $VR \leftarrow$  FindBestValue_Rel ( $r, [R_i, R\text{-type}, [A1, A2]],$ 
      BasicLevelValue_Rel ( $R\text{-type}$ ),  $RemDist$ )
     $OutRuledRefs \leftarrow$  RulesOut_Rel ( $RemDist, [R_i, VR, [A1, A2]]$ )
     $tree' \leftarrow$  UpdateTree_Rel ( $tree, [R_i, VR, [A1, A2]], r$ )
    if  $OutRuledRefs \neq \{\}$   $\wedge$   $tree' \neq$  nil
    then  $tree \leftarrow tree'$ 
       $RemDist \leftarrow RemDist - OutRuledRefs$ 
      if  $r = A1$  *
      then  $r' \leftarrow A2$  *
      else  $r' \leftarrow A1$  *
      endif *
       $tree2 \leftarrow$  MakeReferringExpression ( $r', s, [R_i, VR, [A1, A2]]$ ) *
       $tree \leftarrow$  AddTree ( $tree, tree2$ )
    endif
    if  $RemDist = \{\}$ 
    then  $tree \leftarrow$  AddDefDet ( $tree$ )
      return  $tree$ 
    endif
  endfor
endfor
 $tree \leftarrow$  AddIndefDet ( $tree$ )
return  $tree$ 
endif

```

FIGURE 4.12: Extension of the modified Incremental Algorithm which incorporates relational descriptions (version 2). A * marks the differences with the first version, shown in Figure 4.8.

```

RulesOut_Rel (RemDist, [R, VR, [A1, A2]], r)
if VR = novalue
then return {}
else if r = A1
then return {x : x ∈ RemDist ∧
              ¬∃y ∈ D : UserKnows (x, [R, VR, [x, y]]) = true }
else return {x : x ∈ RemDist ∧
             ¬∃y ∈ D : UserKnows ([R, VR, [y, x]]) = true }
endif
endif

```

FIGURE 4.13: RulesOut_Rel (version 2).

and the set *RemDist* is empty from the start. This means that after the first property of h_1 (`< type, doghouse >`) has been included in the description, the selection of properties is finished: h_1 is the most salient entity in the domain which contains something (i.e., is the second argument in an in relation). A definite determiner is added to the tree for h_1 , producing the description *the doghouse*. The algorithm now returns to the initial call of `MakeReferringExpression`. The tree for h_1 is inserted in the empty NP position in the tree for d_2 . Since d_2 has no more remaining distractors, the selection of relations is finished. A definite determiner is inserted, and the description *the dog in the doghouse* is returned. The interesting thing about this description is that it is distinguishing, while neither *the dog* nor *the doghouse* in isolation are.

Discussion

Let us take stock. It has been shown that some relatively simple modifications to the (modified) Incremental Algorithm may suffice for the generation of descriptions involving relations. This exercise shows how insights of Dale and Haddock 1991 can be incorporated in Dale and Reiter's Incremental Algorithm, which is more efficient than the Greedy Heuristics strategy used by Dale and Haddock. In contrast to Horacek 1997 and Stone and Webber 1998, this algorithm is fully explicit about which properties should be tried in which order. The use of subsumption hierarchies on relations seems to offer an attractive and plausible means of obtaining some of the required flexibility.

To deal with more complex domains than were discussed in this section, further finetuning of the algorithm is still necessary. For instance, in a domain where different doghouses contain different types of dogs, it is necessary to distinguish a doghouse containing a mastiff from a doghouse containing, say, a chihuahua. To allow for such distinctions, the function `RulesOut_Rel` needs to take the properties of the relatum r' into

account when dealing with a relation \mathcal{R} . This would enable the function to rule out distractors that have a similar relation (of the same datatype and relation-type as \mathcal{R}) with an entity y , where y is known to have different properties from r . This is only one of many improvements that can still be made to the algorithm, in order to deal with the generation of relational descriptions in increasingly more complex situations.

Finally, it should be stressed that in discussing the generation of relational descriptions several simplifying assumptions have been made, for instance concerning the ordering of relations, the set-up of the database and the subsumption hierarchy of types of spatial relations. These issues should be further investigated in future work. However, none of them seems to call for drastic changes to the proposals made in this section.

4.5.3 Bridging descriptions

The two extensions of the modified Incremental Algorithm described above and displayed in Figures 4.5 and 4.12 respectively can be combined in a straightforward way and this combination allows for the generation of bridging descriptions. From the current perspective, a bridging description is just a relational description with a highly salient relatum. To illustrate this, consider a domain of discourse which contains three entities d_1 , d_2 and d_3 : d_1 is a man, while d_2 and d_3 are chihuahua's of the same size and colour, the former being in the possession of d_1 , the latter being a stray dog. Suppose that the man has just been mentioned (*A man is walking in the park*) and thus is maximally salient. Now the algorithm attempts to generate a description of d_2 . To begin with, the type (dog) is included. The attributes size and colour are not included in the description since they fail to rule out the other chihuahua. Finally, the possessive relation is encountered: the fact that d_2 is in the possession of d_1 is included as this does rule out the stray chihuahua. At this point, the algorithm enters the recursion. Since d_1 is the single most salient entity in the domain, and has been referred to in the previous sentence (thus providing an appropriate antecedent), the reference to d_1 can be pronominalised, and a suitable pronoun is inserted in the current tree. Normally, this would result in (a tree for) *dog of him*, but following common practice (see e.g., Geurts 1995, Krahmer and van Deemter 1998) the function `UpdateTree` rewrites such descriptions using a possessive pronoun as determiner, with *his dog* as the net-result. As a rule of thumb, it is assumed that this happens only if the relatum is animate. Thus, if a particular car c is highly salient and reference is made to the motor of c , the resulting NP will not be *its motor* but *the motor*. It should be noted that this distinction is highly language dependent. In

French, for example, it is common to refer to someone's hand as *la main* instead of *sa main*.

4.6 Experiment

Some of the hypotheses underlying the modified Incremental Algorithm have been experimentally tested using a forced choice experiment, described in Verhagen 1999. In this experiment, the subjects had to indicate for a number of texts which of two versions of these texts they found most natural. The two versions differed with respect to the description of one item. In this section, the experiment is described and its results are discussed.

4.6.1 Hypotheses and assumptions

The following hypotheses were tested in the experiment:

Hypothesis I: People prefer anaphoric descriptions that contain fewer properties than their antecedents.

Hypothesis II: People prefer anaphoric descriptions that express attribute values which are closer to the basic level value than those expressed in their antecedents.

Hypothesis III: If an anaphoric expression can refer to only one entity, people prefer the anaphor to be pronominalised.

Hypothesis IV: Hypotheses I and II apply also if there is one intervening sentence between the anaphor and its closest antecedent. (This hypothesis is divided into two parts: IVa relates to Hypothesis I, and IVb relates to Hypothesis II.)

The first three hypotheses address the question if people really prefer the reduced, generalised or pronominalised descriptions generated by the modified algorithm over the full descriptions that would be generated if the effects of salience were not taken into account. The fourth hypothesis tests if a discourse entity still remains salient after it has not been mentioned for one sentence (the gradual decrease in salience discussed in Section 4.4).

In the experiment, the most basic form of salience weight determination was used, simply assuming that an entity that has been mentioned becomes more salient than other non-mentioned entities. This was sufficient for the short texts used in the experiment, some examples of which are shown in Figure 4.14.

Hypothesis I: omission of properties

De poedel en de grote terrier maakten ruzie over een bot.
 { *De terrier / de grote terrier* } *ging er met het bot vandoor.*
 The poodle and the large terrier had a fight over a bone.
 { The terrier / the large terrier } ran off with the bone.

Hypothesis II: more basic values

De poedel en de kat woonden op de boerderij.
 { *De hond / de poedel* } *waakte over de kippen.*
 The poodle and the cat were living at the farm.
 { The dog / the poodle } guarded the chickens.

Hypothesis III: pronominalisation

De zwarte poedel had honger.
 { *Hij / de hond* } *ging op zoek naar een bot in de keuken.*
 The black poodle was hungry.
 { It / the dog } went to the kitchen to look for a bone.

Hypothesis IVa: intervening sentences (I)

John kocht een grote witte chihuahua en een poedel in het asiel.
De poedel was een koopje.
 { *De chihuahua / de grote witte chihuahua* } *was iets duurder.*
 John bought a large white chihuahua and a poodle at the asylum.
 The poodle was a bargain.
 { The chihuahua / the large white chihuahua } was a bit more expensive.

Hypothesis IVb: intervening sentences (II)

De bruine rottweiler en de zwarte rottweiler lagen in de tuin te slapen.
Plots begonnen twee grijze katten te vechten in de tuin.
 { *De bruine hond / de bruine rottweiler* } *werd wakker door het lawaai.*
 The brown rottweiler and the black rottweiler were sleeping in the garden.
 Suddenly, two grey cats started fighting in the garden.
 { The brown dog / the brown rottweiler } was awakened by the noise.

FIGURE 4.14: Examples of texts associated with Hypotheses I to IV.

4.6.2 Method

The modified algorithm is meant to generate referring expressions which will be preferred by human hearers, rather than to reproduce the variety of referring expressions which human speakers produce in various contexts. Therefore a forced-choice experiment was used to test the hypotheses presented in Section 4.6.1, rather than a production experiment. In the experiment, the subjects had to indicate their preference for one of two alternative referring expressions.

The experiment was performed by 51 naive subjects. All subjects were native speakers of Dutch, except one, whose mother tongue was English. The subjects (of different ages and backgrounds) were presented with 32 texts in Dutch, displayed in two versions on a computer screen. For each text, they had to indicate which of its two versions they found most natural.

4.6.3 Materials

The texts used in the experiment were constructed to test the hypotheses presented above. There were eight texts per hypothesis. (In the case of Hypothesis IV, four of these texts were associated with IVa and four texts were associated with IVb.) The texts were written in Dutch and consisted of two or three sentences. Each text had two versions, A and B, that were the same except for the description of one discourse item (the grammatical subject in the final sentence). The description in the A version was in line with the hypothesis associated with the text, whereas the description in the B version was overspecified according to the same hypothesis. Both descriptions were distinguishing.

Figure 4.14 shows an example text for each of the hypotheses. To save space, only one text is shown with the pair of differing descriptions in the last line. The first description of each pair is the description that is in line with the corresponding hypothesis, and the second one is the overspecified description.

4.6.4 Results

Table 4.1 shows the results of the experiment. For each hypothesis, it shows the number of texts for which (i) a significant number of subjects preferred version A; (ii) a significant number of subjects preferred version B; (iii) there was no significant preference for either version. Significance was computed using the χ^2 test.

The results of the experiment can be summed up as follows. For all texts associated with Hypotheses I and III, a highly significant number of subjects ($p < 0.001$) preferred version A, containing a description in line with the associated hypothesis. However, for only two of the texts asso-

	Significant preference for A	Significant preference for B	No significant preference
Hypothesis I	8 ^a	0	0
Hypothesis II	2 ^b	5 ^c	1 ^d
Hypothesis III	8 ^a	0	0
Hypothesis IV	4 ^a	3 ^a	1 ^e

a: $p < 0.001$

b: for one text, $p < 0.005$ and for the other text $p < 0.001$

c: for one text, $p < 0.01$ and for the other four $p < 0.001$

d: non-significant preference for A ($0.1 < p < 0.25$)

e: non-significant preference for B ($0.25 < p < 0.5$)

TABLE 4.1: The number of texts for each hypothesis, for which (i) a significant number of subjects preferred version A; (ii) a significant number of subjects preferred version B; (iii) there was no significant preference for either version.

ciated with Hypothesis II, the subjects had a significant preference for version A. For five of the texts associated with this hypothesis, there was a significant preference for version B. Hypothesis IV also showed mixed results: for half of the texts associated with this hypothesis, version A was significantly preferred, and for three other texts, B was significantly preferred. This difference in preference corresponds closely to the division of Hypothesis IV into parts IVa and IVb. As shown in Table 4.2, for the texts associated with Hypothesis IVa the A version is significantly preferred in three out of four cases, whereas for the texts associated with Hypothesis IVb it is the B version which is significantly preferred in three out of four cases.

	Significant preference for A	Significant preference for B	No significant preference
Hypothesis IVa	3 ^a	0	1 ^b
Hypothesis IVb	1 ^a	3 ^a	0

a: $p < 0,001$

b: non-significant preference for B ($0,25 < p < 0,5$)

TABLE 4.2: The results for Hypothesis IV, divided into two groups of text pairs, corresponding to Hypothesis I and Hypothesis II respectively.

4.6.5 Discussion

The experimental results confirm Hypotheses I and III: the subjects in the experiment showed a significant preference for anaphoric descriptions

that contained fewer properties than the antecedent, or that were pronominalised. A plausible explanation for this preference is that the reduced descriptions increased the coherence of the example texts, without giving rise to ambiguity. The findings for Hypothesis III are in line with the experimental results of Gordon et al. 1993, who found that utterances are more difficult to read if they contain a definite description or a proper name where a pronoun could have been used.

Hypothesis II was not confirmed: for only two of the eight texts, a significant number of subjects preferred the version containing a description with a more basic head noun than the antecedent. For five texts, a significant number of subjects preferred the version that did *not* contain a description with a more basic head noun. When looking at the three²⁰ texts for which a more basic head noun was preferred (in line with Hypothesis II), we see that in all three cases, the head noun in the anaphor is a substring of the more specific head noun of the antecedent, e.g., *siamese cat* → *cat*. In the other five texts, this is not the case; here the more basic head noun is completely different from the more specific one. An example is *poodle* → *dog*. These observations suggest that the preference for repeating the same head noun, observed in five of the eight texts associated with Hypothesis II, should be seen as a preference for using the same *wording* in both anaphor and antecedent. This is similar to the ‘priming’ effect that has been found in dialogues: speakers tend to use the same literal expressions as their interlocutors (see e.g., Levelt and Kelter 1982 and Clark and Wilkes-Gibbs 1986). Presumably, holding on to the same wording (both within and among speakers) should be seen as another method for maintaining the coherence of a discourse. When the more basic head noun is lexicalised as a substring of the more specific one (*cat* versus *siamese cat*), then there is no ‘switch’ to a different wording. In such cases, the use of the more basic head noun is preferred.²¹

Like Hypothesis II, Hypothesis IVb was not supported by the experimental results. This hypothesis was constructed to test if Hypothesis II holds when there is an intervening sentence between anaphor and antecedent. For three of the four texts associated with IVb, the B version was preferred. This may be explained through the fact that the A versions of these three texts contained a head noun with a different wording than the antecedent. As discussed above, Hypothesis II does not hold for such head nouns. For the fourth text, the A version was

²⁰This includes one text for which a non-significant majority of the subjects preferred version A; see Table 4.1.

²¹Possibly, the subjects perceived these cases as the omission of a property (Hypothesis I) rather than the use of a different head noun (Hypothesis II).

preferred. In this text, the more basic head noun used in the anaphor had the same wording as the antecedent. Thus, Hypothesis IVb shows exactly the same picture as Hypothesis II. On the other hand, for three out of four texts associated with Hypothesis IVa (constructed to test if Hypothesis I holds when there is an intervening sentence), version A was preferred. This is in line with the results for Hypothesis I.

The conclusions that can be drawn from the experiment are the following.²² People prefer anaphoric descriptions that contain fewer properties than the antecedent (provided that the referent is the most salient entity with the included properties); this holds even when there is an intervening sentence between anaphor and antecedent. So, the strategy of not including non-discriminating properties is correct, as is the gradual decrease of salience argued for in Section 4.4. In addition, people prefer pronominalised anaphoric descriptions over non-pronominalised ones (provided that the referent is the most salient discourse entity). This justifies the extension of the modified algorithm to generate pronouns, discussed in Section 4.5.1.

Finally, people prefer to use a more basic head noun in the anaphor only if the wording is similar to that of the antecedent; otherwise, they prefer to repeat the same head noun even if this leads to overspecification. Apparently, using the same wording in both antecedent and anaphor leads to a higher degree of perceived coherence than reducing overspecification through the use of a more general head noun. This finding suggests that an additional check should be made in the `FindBestValue` function of the modified algorithm: a more specific value (*new-value*) should be chosen either if it rules out more distractors than the more general value (*value*), or if its words are ‘better matching’ than the more general value. To check this, a `BetterMatch` function is added, which returns `true` if (i) the intended referent *r* has an antecedent *c*, and (ii), *new-value* has the same wording as the corresponding value in *c*, while *value* has a different wording. To check the latter condition, a function `SameWording` is used which takes as input the antecedent *c* and an attribute value pair $\langle A, V \rangle$ and returns `false` if the linguistic realisation of $\langle A, V \rangle$ is not a substring of the corresponding expression in *c*. In the experiment, Hypothesis II was only tested for the `type` attribute, as this attribute is generally expressed as the head noun in a description. However, we assume that the effect of wording also holds for other

²²A qualification is in order here, since the experiment was based on written, not spoken, texts. Therefore, it does not necessarily follow from the experiment that people have the same preferences concerning *spoken* language. It may be the case that people have a higher preference for overspecification in spoken language, as limitations of short term memory play a more important role there.

attributes, such as colour. For instance, a cat that has been previously described as *the chestnut cat* will not be anaphorically referred to as *the brown cat*. The extended FindBestValue function is shown in Figure 4.15.

```

FindBestValue (r, A, initial-value, RemDist)
if UserKnows (r, ⟨A, initial-value⟩) = true
then value ← initial-value
else value ← novalue
endif
if (msv ← MoreSpecificValue(r, A, value)) ≠ novalue ∧
    (nv ← FindBestValue (r, A, msv, RemDist)) ≠ novalue ∧
    (| RulesOut (RemDist, ⟨A, nv⟩)| > | RulesOut (RemDist, ⟨A, value⟩)| ∨
     BetterMatch (r, ⟨A, nv⟩, ⟨A, value⟩) = true)
then value ← nv
endif
return value

BetterMatch (r, ⟨A, nv⟩, ⟨A, value⟩)
if ∃c : Antecedent(c, r) ∧
    SameWording (⟨A, value⟩, c) = false ∧
    SameWording (⟨A, nv⟩, c) = true
then return true
else return false
endif

```

FIGURE 4.15: FindBestValue extended with a check on wording.

Finally, a brief example can illustrate the effect of the proposed extension. Assume that the domain of discourse contains a cat, a chihuahua and a poodle, and that the first two entities have been referred to as *the cat* and *the chihuahua* respectively. (The poodle has not been referred to.) Now a second reference to the chihuahua is generated. When determining the best value for its type attribute, it turns out that the values *dog* and *chihuahua* rule out the same number of distractors. (Both rule out the cat.) In the original FindBestValue function, the value *dog* would be chosen, since it is more general than *chihuahua*. Using the extended function from Figure 4.15, however, the algorithm would find that the more specific value *chihuahua* provides a better match with the antecedent than *dog*, since the expression for *chihuahua* (*chihuahua*) is a substring of the expression for the type value in the antecedent, whereas the expression for *dog* is not. So the algorithm will return the description *the chihuahua* and not *the dog*.

4.7 Integration of the modified algorithm in the LGM

This section deals with the embedding of the modified algorithm in the LGM. First, it is briefly shown how referring expressions can be generated in the LGM using the new `MakeReferringExpression` algorithm. Then some issues concerning the accentuation of the generated expressions are discussed.

4.7.1 Generation of referring expressions

The new `MakeReferringExpression` algorithm proposed in this chapter has been implemented and embedded in the LGM as a replacement of the original `Express` function used in the LGM to generate referring expressions.²³ As discussed in Chapter 2, and briefly noted in the introduction to this chapter, in the original LGM definite descriptions were generated in a rather *ad hoc* fashion. In addition, a generate-and-test strategy was employed to select expressions for specific entities. In trying to find an appropriate expression to refer to an entity, the original `Express` function generated a set of possible referring expressions, consisting of at least a pronoun, one or more definite descriptions, and a proper name (if available). Linguistically inappropriate expressions were then filtered out from this set on the basis of local (Binding Theory) and global discourse constraints (presence of suitable antecedents). From the remaining expressions, one was chosen at random. This somewhat inefficient strategy can be discarded when the new `MakeReferringExpression` algorithm is embedded in the LGM. The `MakeReferringExpression` algorithm only produces one description (in the version of the algorithm discussed in Section 4.5.1, this can be either a pronoun, a definite or an indefinite description). Since this description is already optimal given the current context, it can be used immediately without the need of additional discourse filters. In short, using `MakeReferringExpression` in the LGM has two distinct advantages. In the first place, referring expressions are generated in a principled manner, and secondly, the selection of referring expressions is simplified considerably.²⁴

Incorporating the `MakeReferringExpression` algorithm requires only a few additions to the LGM. The most important one is the addition of a salience function. This can be done in a fairly straightforward fashion, as all information required to compute salience according to Defini-

²³The implementation was carried out by John-Pierre Verhagen, see Verhagen 1999.

²⁴For entities that have a proper name available, the only choice that still has to be made outside the algorithm is whether to use the expression generated by the algorithm or the proper name. Presumably, proper names are appropriate when introducing an entity, but not in situations where an anaphoric description is allowed. This issue is not addressed here.

ition 5 from Section 4.4 is already present in the LGM: the Discourse Model records which entities have been mentioned in each utterance, thus providing the forward looking centers of each utterance, and the syntactic templates used in the LGM provide the syntactic information needed to determine the ordering of these centers.

The domain database used in the LGM (see Section 2.4.1) provides information about the properties of the domain entities that can be used by the `MakeReferringExpression` algorithm. The set-up of this database, which is used in the LGM to record ‘background’ information on the entities in the domain, can remain unchanged. In the discussion of `MakeReferringExpression`, it was assumed that the user (hearer) is aware of all the information stored in the domain database. This corresponds to the way the domain data are currently used in the LGM: they are assumed to be static and most of them are supposed to be known to the user, so they can be used in descriptions of discourse entities. For instance, in `GoalGetter` the football team PSV can be referred to as *the team from Eindhoven*, on the basis of the attribute-value pair $\langle \text{place, Eindhoven} \rangle$ that is stored for PSV in the database. Nevertheless, the modified algorithm has not been implemented in any of the existing LGM-based applications, such as `DYD` or `GoalGetter`, since they all have proper names available for the entities in their domain, and since – due to the character of the generated texts – definite descriptions are only used occasionally in these systems. (For instance, in the texts generated by `GoalGetter` most discourse entities are mentioned non-anaphorically, in which case a proper name is usually the most suitable type of expression.) As a consequence, the existing LGM applications do not provide an ideal environment for using (and testing) the modified algorithm. For this purpose, the `MakeReferringExpression` algorithm has been implemented in a new (toy) LGM application called `Pavlov`. This application generates short (one-paragraph) texts in a cats and dogs domain.

4.7.2 Referring expressions and accentuation

Accentuation of the referring expressions generated by the modified algorithm is taken care of by the general LGM accentuation rules in the prosody module. In short, referring expressions are accented in principle, except when they are ‘object-given’ and not contrastive. A phrase is object-given if it refers to an entity that has been mentioned previously in the same paragraph. In addition, words occurring in a referring expression may be deaccented due to concept-givenness. A word is concept-given if it expresses a concept that subsumes, or is identical to, a concept expressed in the same or previous sentence. In example

(14), the word *poodle* in the second sentence is concept-given since the same word (expressing the same concept) has been used in the previous sentence. The word *winner* is concept-given as well, because the verb *won* in the previous sentence has already evoked the concept ‘winning’.

- (14) The white poodle won the first prize in the hair contest.
 The grey poodle was the winner for obedience.

Contrastive accentuation of referring expressions works as explained in Section 3.6: if two referring expressions express contrastive data structures (i.e., they refer to entities that occur as alternative items in two contrastive sentences), then the words expressing their contrastive attributes receive contrastive accent. This can also be illustrated using example (14). Assume that the domain consists of many dogs, including the two dogs of which the data structures are given in Figure 4.16, and that the two sentences in (14) have been generated to express the data given in Figure 4.17.

$\left[\begin{array}{l} \textit{animal} \\ \text{referent : } d_{26} \\ \text{type : } \textit{poodle} \\ \text{size : } \textit{large} \\ \text{colour : } \textit{white} \end{array} \right]$	$\left[\begin{array}{l} \textit{animal} \\ \text{referent : } d_{43} \\ \text{type : } \textit{poodle} \\ \text{size : } \textit{large} \\ \text{colour : } \textit{grey} \end{array} \right]$
--	---

FIGURE 4.16: Data structures for the entities mentioned in (14).

$\left[\begin{array}{l} \left[\begin{array}{l} \textit{win_event} \\ \text{winner : } d_{26} \\ \text{contest : } \textit{hair} \\ \text{prize : } \textit{first} \end{array} \right] \\ \\ \left[\begin{array}{l} \textit{win_event} \\ \text{winner : } d_{43} \\ \text{contest : } \textit{obedience} \\ \text{prize : } \textit{first} \end{array} \right] \end{array} \right]$
--

FIGURE 4.17: Example input data for (14).

The two data structures shown in Figure 4.17 are contrastive, because they are of the same type (*win_event*) but have different attribute values for their *winner* and *contest* attributes. The phrases expressing those values (in the second sentence) should therefore receive contrastive

accent. In Figure 4.17, the values for the winner attributes are shown as indices, which abbreviate the data structures shown in Figure 4.16. These two data structures are contrastive too: they are of the same type (animal) but have different values for their colour attributes. This means that the contrastive accent, assigned to the expression for d_{43} should fall on the word *grey*, expressing the colour attribute. Given that the words *poodle* and *winner* must be deaccented due to concept-givenness, the second sentence of (14) is thus accented as follows, with contrastive accents on *grey* and *obedience*.

(15) The GREY poodle was the winner for OBEDIENCE.

The accentuation of relational descriptions is expected to follow along the same lines. For instance, if the expressions *the dog left of the doghouse* and *the dog right of the doghouse* occur in consecutive, contrastive sentences, the second preposition (*right*) receives contrastive accent, resulting in the accentuation pattern *the dog RIGHT of the doghouse*.²⁵

An interesting question is whether or not bridging descriptions should be accented. General wisdom has it that new information should receive a pitch accent, while given information should not. Bridging descriptions do not fit easily in this picture. Prince 1981 places them in between given and new in her givenness hierarchy. Birner 1994, however, uses inversion data to argue that they are ‘discourse old’, and Williams 1998 indeed always deaccents them. It should be stressed, however, that bridging descriptions themselves do not form a homogeneous category (Clark 1975; see also Piwek and Krahmer 2000). This is confirmed by Hitzeman (p.c.) who claims that there is no clear connection between bridging descriptions and the presence or absence of accents.

4.8 Summary

This chapter has discussed a generalisation of Dale and Reiter’s Incremental Algorithm which extends the original algorithm in a number of respects. To begin with, the notion of context sets has been made more precise by adding salience weights. This makes it possible to generate descriptions in a fully context-sensitive manner, without jeopardising the attractive properties of the original algorithm. Additionally, the algorithm now immediately attempts to incorporate selected properties in the NP tree under construction.

Three related extensions of the modified algorithm have been de-

²⁵If the expressions do not occur in contrastive sentences, the same effect is achieved if *right* is accented due to newness and the other words in the description are deaccented due to concept-givenness. This is the ‘default accent’ effect pointed out by Ladd 1980:81.

scribed. First, a simple pronominalisation decision within the algorithm was discussed. Second, it has been shown that some modifications of the (modified) Incremental Algorithm allow for the generation of relational descriptions. The combination of these two extensions enables the generation of bridging descriptions.

An experiment has been carried out to test some of the basic assumptions underlying the modified algorithm. Most of these assumptions were confirmed, except for the assumption that people prefer to use anaphors containing a more general head noun than their antecedent. Such a preference was only observed for those cases where the more general head noun had a similar wording to the more specific head noun used in the antecedent. In the other cases, repetition of the same (specific) head noun was preferred. It has been shown how the modified algorithm can be adapted to take this result into account.

Finally, the modified Incremental Algorithm has been implemented and integrated in the LGM. The method for contrastive accent assignment described in Chapter 3 can be used for accent assignment in the descriptions produced by the algorithm.

Language generation in a spoken dialogue system

5.1 Introduction

So far, language generation has been discussed in the context of spoken monologues, where the system speaks and the user listens to the information that is presented by the system. However, progress in speech technology has caused an increased interest in spoken *dialogue* systems, where the user is not only spoken to but can also speak to the system.

A relatively common type of spoken dialogue system is the *information system*, where the system functions as a mediator between the user and a database. In the dialogue between user and system, the system prompts the user to specify which database information he or she is interested in, and then retrieves and presents the required information. Application domains of information systems include travel planning (ATIS, Seneff et al. 1991; TOOT, Walker et al. 1998), theatre ticket reservation (SCHISMA, van der Hoeven et al. 1996), and weather reports (Jupiter, Zue et al. 2000).

In this chapter, a Dutch public transport information system called OVIS is discussed. In OVIS, the D2S system is used for the generation of spoken output. Most existing dialogue systems do not employ such a sophisticated method for output generation, which may be partially explained by the fact that they have a central dialogue management module, which takes over some of the ‘standard’ language generation tasks. It is the dialogue manager which decides what to say and when to say it, thus in essence performing the first two of the generation tasks discussed in Section 1.2.1: *content determination* and *discourse planning*. The responsibility of language generation is restricted to the

formulation of the system utterances ('how to say it').¹ Formulation includes the four remaining generation tasks as discussed in Section 1.2.1. However, in most spoken dialogue systems several of these can be performed in a rather limited way. Since dialogue prompts generally consist of one sentence only (usually a question), *sentence aggregation* is hardly necessary. In addition, *lexicalisation* and the generation of *referring expressions* generally require hardly any linguistic knowledge, because most systems operate in restricted domains where fixed expressions are common. This leaves *linguistic realisation* as the principal task to be performed for prompt generation. This task can generally be carried out using a simple string manipulation approach, because the dialogue domain is usually so restricted that it is possible to list all desired output strings (after abstracting over variable information). All in all, in many existing dialogue systems there appears to be no need for a full-fledged language generation module. In fact, several systems do not even distinguish a separate language generation module. Examples of this are the public transport information system described in Aust et al. 1995 and the Swedish Waxholm system (Blomberg et al. 1993).

The above limitations hold in particular for the generation of dialogue prompts. For the presentation of database information, things are slightly different, since this generally involves the generation of texts that are longer and more complex than the average dialogue prompt. This is more like monologue generation, and most of the regular generation tasks *do* need to be carried out. Still, in many current dialogue systems the information presentations are relatively short and have a fixed structure, which means that simple application-specific methods are sufficient here as well.

In sum, for most current dialogue systems, relatively simple techniques suffice for the adequate production of language output. However, as research on other aspects of spoken dialogue systems progresses, the demands on language generation will increase (Cole et al. 1995a, Zue 1997). The growing robustness of speech recognition methods, which are used to recognise the dialogue contributions of the user, allows for increasingly larger domains of discourse, and advances in dialogue management strategies enable dialogue managers to handle more complex conversations. This means that the variety and complexity of system utterances will increase, and more general language generation methods, based on linguistic insights, will be required. This chapter anticipates on this development, by not only discussing language gener-

¹This task division corresponds to the division between conceptualisation and formulation in human sentence production (see e.g., Kempen 1977, Levelt 1989).

ation in an existing dialogue system, OVIS, but also by looking ahead to language generation in more sophisticated dialogues.

The language generation module for the OVIS system is discussed in Section 5.2. In OVIS, referring expression generation and accent assignment can be performed by simpler techniques than those discussed in Chapters 3 and 4, but – as discussed above – these techniques will probably not suffice in future dialogue systems covering more complex domains. Therefore, Section 5.3 examines how a dialogue context may influence the performance of these two tasks. In this section it is shown that in a dialogue context, determining information status (which is relevant for both tasks) is less straightforward than in monologue generation: due to possible communication problems, something that is said by one speaker cannot be assumed to be immediately known by the other. Section 5.4 then discusses which linguistic and prosodic cues from the user can help the system to determine information status in human-system interaction.

5.2 Using the LGM in OVIS

The D2S system is currently being used for output generation in a Dutch spoken dialogue system called OVIS, which is an abbreviation of *Openbaar Vervoer Informatie Systeem* (“Public Transport Information System”). The OVIS system is accessible via the telephone and provides the user with spoken information concerning train connections and ticket prices.

This section starts with a general overview of the OVIS architecture and of the tasks performed by its different components (Section 5.2.1). Then the use of the LGM in OVIS is discussed, focusing on the differences with monologue generation and the consequences for the LGM. First, the input and output of the LGM in OVIS are described (Section 5.2.2), followed by a discussion of the syntactic templates used in OVIS (Section 5.2.3), and of coherence and variation in the OVIS prompts (Sections 5.2.4 and 5.2.5 respectively). The section ends with a discussion.

5.2.1 The OVIS architecture

The architecture of OVIS, which is similar to that of most spoken dialogue systems (see e.g., Cole et al. 1995b), is shown in Figure 5.1. The system has the following modules:

Speech recognition. This module processes the speech signal produced by the user, in order to determine which words have been spoken. The occurrence of acoustic and phonetic variability in the speech signal

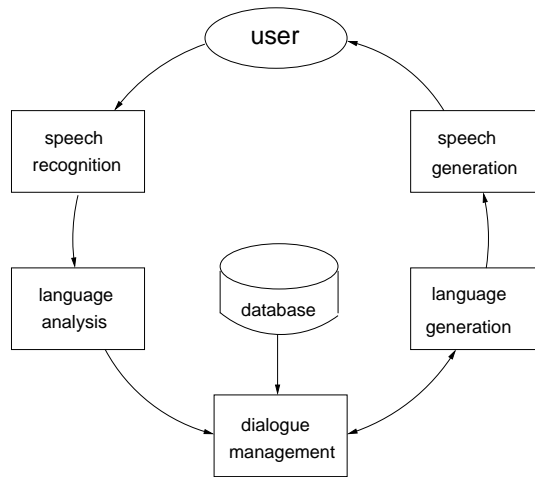


FIGURE 5.1: OVIS architecture.

make this a difficult task, especially since the OVIS domain has a large vocabulary with many similar-sounding words (e.g., the train stations Baarn and Maarn). Misrecognitions are therefore unavoidable. Speech recognition in OVIS (Strik et al. 1996) is based on a statistical technique called Hidden Markov Modelling (see Jelinek 1997 for a discussion). The output of the speech recognition module is a word graph, i.e., a lattice of word hypotheses.

Language analysis. This module determines which of the word sequence hypotheses returned by speech recognition is the most likely one to have actually occurred, and converts this word string into a meaning representation. In OVIS, two alternative language analysis modules are available. The first module (van Noord et al. 1999) employs a grammar, which encodes information about the syntactic and semantic constraints on possible sentences in the OVIS domain. The second module (Bonnema et al. 1997, Bod 1998) employs a statistical approach, comparing (parts of) an input string to subtrees in a large annotated corpus of example utterances. Both language analysis modules return the same type of meaning representation of the user's utterance.

Dialogue management. This module is responsible for coordinating the interaction with the user and consulting the database. On the basis of the information about the user's utterance, it determines which

message the system should convey next. Two basic message classes can be distinguished: (i) dialogue prompts, which are part of the dialogue in which the system tries to determine the wishes of the user, and (ii) presentations of information from the database (either price or train time table information). The strategy for deciding which message is the most appropriate at which point in the dialogue is described in Veldhuijzen van Zanten 1998. The output of the dialogue manager is a specification of the type and content of the message that must be expressed.

Language generation. This module is the LGM from D2S. It transforms the abstract message specification sent by the dialogue manager into a natural language utterance. Since the dialogue manager has already performed the task of deciding when to say what, the LGM is only responsible for the formulation of the message. Another difference with monologue generation is that each system message should be expressed within one dialogue turn. Only one interrogative sentence may be generated in each turn, and at the same time, the complete message from the Dialogue Manager should be expressed. Since it is not always possible to fulfil this last requirement, the LGM not only produces an enriched text which is passed on to speech generation, but also provides feedback to the Dialogue Manager, indicating which parts of the message have been expressed.

Speech generation. Finally, the speech generation module (the SGM from D2S, see Section 2.6) transforms the enriched text string it receives from language generation into a speech signal. In OVIS, both available speech generation methods (advanced phrase concatenation and diphone synthesis) have been used, in different versions of the system.

5.2.2 Input and output of the LGM

The domain of conversation in OVIS is specified in a so-called ‘frame structure’ that is used by the dialogue manager (from now on abbreviated as DM). This frame structure consists of a typed hierarchy of slots and values, representing the information which can be talked about by the user and the system. (For a full specification of the frame structure see Veldhuijzen van Zanten 1999a.) The data structure which is the basis for generation in the LGM closely corresponds to the frame structure used by the DM. Figure 5.2 shows an important part of the OVIS data structure: the value of its info field. This field contains all the data that are required to retrieve train time table or price information

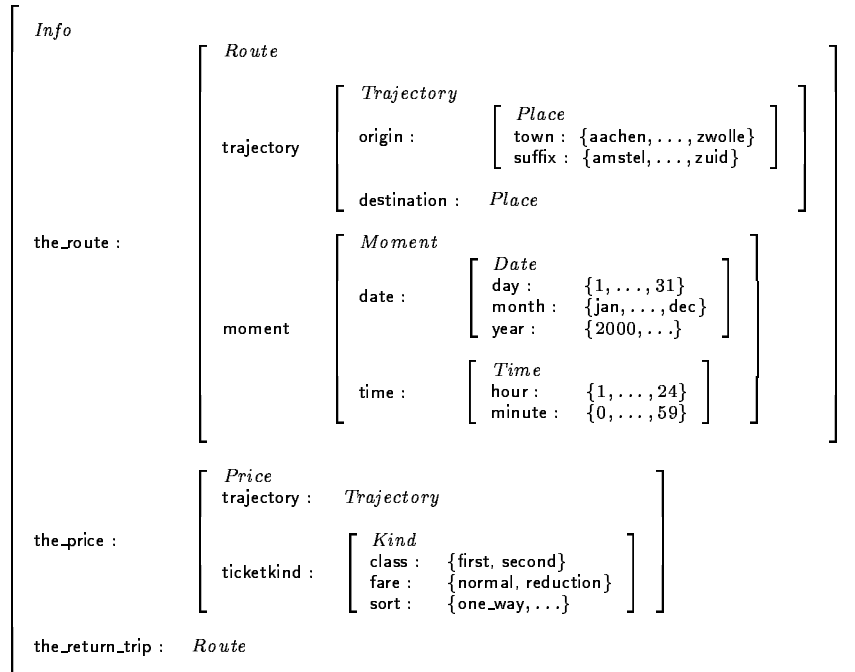


FIGURE 5.2: The info slot from the data structure.

from the database. The desired values within this field are obtained from the user.

The input of the LGM consists of a message from the DM, specifying which values must be added to the data structure and which actions must be performed on its fields and values. As was mentioned in the previous section, there are two basic message classes: dialogue prompts and information presentations. Here, only the generation of the dialogue prompts is discussed, since information presentation in OVIS basically corresponds to monologue generation as discussed in the previous chapters. (See also Theune 2000.)

There are two general types of dialogue prompts: *requests for information* on the user's wishes, where the system asks the user for the origin, destination and moment of his or her intended journey, and *dialogue meta-acts*, utterances about the dialogue itself which may be used to deal with communication problems. The most important meta-act is *verification*: due to the imperfect speech recognition, the system con-

stantly has to verify whether it has understood the user's utterances correctly.

Three types of information requests are distinguished in OVIS: *open questions* for the value of a specific field in the data structure; *alternatives questions*, where the user is offered a choice between several possible values; and *yes-no questions* concerning one possible value. Dialogue meta-acts may take the form of assertions, for instance when the system informs the user of an inconsistency,² or of (possibly elliptic) yes-no questions, in the case of verifications. The formal language in which the messages from the DM are expressed is described in Veldhuijzen van Zanten 1999b;2000; the same language is used for the feedback messages from the LGM to the DM. In (1) and (2) below, a few example messages from the DM are given, together with an informal description (in italics) and the corresponding system utterance generated by the LGM. The system utterances are given in translation.

(1) Information requests:

DM: `user.wants.info.the_route.trajectory.origin ??`
 (*Open question for the departure station*)
 LGM: From where do you want to leave?

DM: `user.wants.info.the_price.class ?! {first | second }`
 (*Alternatives question for the travel class*)
 LGM: Do you want to travel first or second class?

DM: `user.wants.info.the_return_trip.moment.date.day ?= 3`
 (*Yes-no question for the day of the return trip*)
 LGM: Do you want to return on the third?

(2) Meta-acts:

DM: `user.wants.info.the_route.trajectory.origin ?+ asd`
 (*Verification question concerning the departure station*)
 LGM: Departure from Amsterdam Central Station?

DM: `user.wants.info.the_route.trajectory.origin.town ** uden`
 (*Noting an inconsistency concerning the town of departure*)
 LGM: Uden does not have a train station.

²For example, when the user wants to travel on the 30th of February.

The messages shown in (1) and (2) are all ‘simple’ in the sense that they only involve one field in the data structure. ‘Complex’ messages are also possible, for instance combining the assertion of an inconsistency with a request for information, as in (3) below. Section 5.2.3 shows some examples where several verifications are combined.

(3) DM: `user.wants.info.the-route.trajectory.origin.town (** uden; ??)`

(Noting an inconsistency concerning the town of origin, and asking an open question for a new value)

LGM: Uden does not have a train station. Please mention another town.

What makes the generation of dialogue prompts substantially different from monologue generation, are the following two restrictions on prompt generation: (i) the LGM should express the complete message from the DM within one system turn, and (ii) a turn may contain only one question. The reason for the latter restriction is that a question is a ‘release-turn’ action, i.e., an action which signifies the end of a turn and indicates that the other dialogue participant may now take the floor (see Traum 1994:61). After a question has been generated, the user is likely to start answering it immediately. Generating more than one question within a turn is therefore likely to lead to interruptions and confusion. In Sections 5.2.3 and 5.2.4, it is shown that the two restrictions mentioned above have made it necessary to adapt the LGM in several ways for use in OVIS.

5.2.3 Syntactic templates

For prompt generation in OVIS, the syntactic templates used in the LGM are extended with an additional element *L*. This element provides information on the ‘level’ of a template, i.e., how many slots the template refers to. Formally, a syntactic template σ in OVIS is a quintuple $\langle S, E, C, T, L \rangle$. Before the new element *L* is discussed, a brief description is given of the use in OVIS of the other template elements.

Many of the syntactic trees *S* in the OVIS templates are minimal, in the sense that only the head of the construction is lexicalised and the gaps coincide with its arguments. This holds in particular for the syntactic templates that are used for the generation of verification questions, which are generally elliptic (e.g., *To Eindhoven?*). Examples are given in Figure 5.3, which shows three OVIS templates that can be used for the generation of elliptic verification questions, only consisting of a PP. In these syntactic templates, only the heads of the PPs are lexicalised.

In comparison to monologue generation, the role of the Express func-

Template DestVerif

$S = \text{To } \langle \mathbf{destination} \rangle?$
 $E = \mathbf{destination} \leftarrow \text{ExpressPlace}(\text{the_route.trajectory.destination})$
 $C = \text{toverified}(\text{the_route.trajectory.destination})$
 $T = \text{'prompts'}$
 $L = 1$

Template TimeVerif

$S = \text{At } \langle \mathbf{time} \rangle?$
 $E = \mathbf{time} \leftarrow \text{ExpressTime}(\text{the_route.moment.time})$
 $C = \text{toverified}(\text{the_route.moment.time})$
 $T = \text{'prompts'}$
 $L = 1$

Template DestTimeVerif

$S = \text{To } \langle \mathbf{destination} \rangle, \text{ at } \langle \mathbf{time} \rangle?$
 $E = \mathbf{destination} \leftarrow \text{ExpressPlace}(\text{the_route.trajectory.destination})$
 $\quad \mathbf{time} \leftarrow \text{ExpressTime}(\text{the_route.moment.time})$
 $C = \text{toverified}(\text{the_route.trajectory.destination}) \wedge$
 $\quad \text{toverified}(\text{the_route.moment.time})$
 $T = \text{'prompts'}$
 $L = 2$

FIGURE 5.3: Abbreviated OVIS templates. The syntactic structures are omitted.

tions E in OVIS is somewhat limited, since in OVIS there is only one way to express most domain entities. The most frequently occurring discourse entities are towns and train stations, and these are always referred to by their proper name, except for some fixed spots in the travel advice where the expression *there* is used. Other frequent expressions in OVIS are times and dates, which do not allow for much variation either. So, the Express functions used in OVIS only return one expression, whereas in monologue systems such as GoalGetter, typically more than one expression is returned. This discrepancy is caused by the different role of variation in OVIS, which is discussed in Section 5.2.5.

The local conditions (C) on the OVIS templates refer to the actions which need to be carried out by the LGM. Information about this is recorded in the LGM's Knowledge State, which has been extended for use in OVIS. In monologue generation, the only action to be performed

is the assertion of information in the data structure. This means that for each part of the data structure, there are only two possibilities: either it has been conveyed to the user (is 'known') or not. The information that is not yet 'known' must be expressed, thus becoming known, and when all information in the data structure has been expressed, generation stops. In a dialogue, however, the situation is rather different. There are many different actions that can be performed on the data structure, as discussed in Section 5.2.2, and the LGM does not know in advance which action has to be performed on which part of the data structure. In OVIS, at the beginning of each system turn the Knowledge State is updated on the basis of the information from the DM, recording which actions must be performed in the current turn. That is, the relevant parts of the data structure receive a label specifying which action needs to be performed on them.³ The conditions on the syntactic templates are then used to check which templates are applicable given the current Knowledge State. For instance, the condition on Template *DestVerif* from Figure 5.3 says that this template is applicable whenever the value of the destination field must be verified. After the template has been applied, the Knowledge State is updated, recording that a verification question about the destination has been asked.

The last of the 'original' template elements, the topic *T*, does not have a real function in the generation of the dialogue prompts. In monologue generation, the topic information serves to group the sentences in the output, but since the OVIS prompts consist of at most two sentences, the content of which is fully determined by the DM, topic information is irrelevant here. Therefore, all syntactic templates that are used for prompt generation simply have the same topic: 'prompts'. The templates that are used for information presentation have another topic, which ensures that after the system has started a presentation, the 'prompt' templates will not be tried until the presentation has finished.

Now we come to the new template element *L*, the *level*, which specifies how many fields of the data structure are dealt with in the template. This information has been added to the syntactic templates because of the restriction, mentioned in Section 5.2.2, that the complete DM message should be expressed using only one question. For the original LGM, this requirement posed a problem in cases where the DM sent instructions concerning more than one field in the data structure. To see why, consider example (4) below. Here, the DM indicates that

³In OVIS, 'Action State' would be a more appropriate name than 'Knowledge State', since it records which actions the system must perform, and which it has performed already.

the values of both destination and time must be verified. In principle, in this situation all three templates shown in Figure 5.3 are applicable. However, we want the LGM to use the template that deals with the highest possible number of fields, instead of making an arbitrary choice from all applicable templates. To achieve this, the level component has been added to the syntactic templates, and the generation algorithm has been adapted so that during generation, higher level templates are preferred over lower level ones. In example (4), Template DestTimeVerif will therefore be chosen for application, as it has a higher level (i.e., deals with more fields) than both other applicable templates. As a consequence, in (4) a question is generated that verifies the values of both destination and time, exactly as specified by the DM.

(4) DM: `user.wants.info.the_route.(trajectory.destination ?+ mas;
moment.time ?+ 10:30)`

LGM: To Maarssen, at ten thirty?

Finally, a note on domain size is in order. The OVIS data structure has almost 60 fields, which may be involved in several dialogue acts, such as open questions, alternatives questions, verifications, etc. The size of this domain makes the template-approach very labour-intensive: it is a lot of work to list all possible utterances and provide them with appropriate syntactic structures and conditions, since there are hundreds of possibilities. The OVIS language generation module currently contains more than four hundred syntactic templates.

5.2.4 Coherence

In principle, the ‘local condition’ approach used in the LGM fits in well with the task division in OVIS, where the LGM carries out the instructions by the DM, and where pre-planning on the part of the LGM is impossible. As shown in Section 5.2.3, only a few additional restrictions on template choice are required to ensure that the entire message can be expressed in one, well-formed system turn. However, for some messages from the DM it is difficult to generate a coherent system turn. An example of such a DM message is given below in (5). Here, the DM instructs the LGM to verify whether the user wants to have route information (as opposed to price information), and whether the user wishes to travel at ten thirty. Combining these two verifications into one verification question would result in an utterance that is either long and awkward, as in (6), or severely underspecified and thus incomprehensible, as in (7).

(5) DM: `user.wants.info.(variant ?+ the_route;
the_route.moment.time ?+ 10:30)`

(6) Do you want to have route information, and do you want to travel at ten thirty?

(7) ?? Route information, at ten thirty?

To ensure coherence of the system utterances, syntactic templates have been constructed for combinations of ‘related’ fields only. Two fields are regarded as related if they share the same mother or grandmother in the data structure; these fields are considered to provide the best combinations for complex verification questions. Examples of related fields are origin and destination (mother: trajectory) and time and destination (grandmother: the_route).

Since the fields involved in (5) do not have the same (grand)mother in the data structure, the LGM does not have one template that can be used for verifying the values of both slots at the same time. Thus, two syntactic templates, each verifying the value of only one of the specified fields, are applicable in the current Knowledge State. First applying one template and then the other, as would be done in ‘monologue mode’ (where generation continues until the set of applicable templates is exhausted), is not an option as it will result in the asking of two verification questions in one turn, as shown below in (8).

(8) LGM: At ten thirty? Route information?

Since these two questions are prosodically separated by a final phrase boundary, the user might assume that the turn ends after the first part of the question, and as a consequence start answering too early. To avoid this, the algorithm has been changed so that it only applies the first template that happens to be selected,⁴ and then stops, having detected that a question has been generated. As a result, only a part of the input message is expressed. The DM is informed of this through the feedback sent by the LGM. This is illustrated in (9) below, where it is assumed that Template TimeVerif from Figure 5.3 is picked to generate a verification question concerning the time of travel. The feedback from the LGM to the DM is given below the generated utterance; it indicates which part of the input message has actually been expressed.

(9) DM: `user.wants.info.(variant ?+ the_route;`
`the_route.moment.time ?+ 10:30)`

LGM: At ten thirty?
`user.wants.info.the_route.moment.time ?+ 10:30`

⁴Since both applicable templates have the same level, the LGM picks one of them at random.

In principle, it is possible that the message sent by the DM cannot be expressed by the LGM at all. This may be the case, for instance, if the DM sends a message which is allowed by the message specification language (Veldhuijzen van Zanten 2000), but which the LGM does not expect because it cannot be expressed in a well-formed utterance. A (hypothetical) example would be a message specifying an alternatives question concerning all possible departure stations, as shown in (10) below (specified using abbreviations for the train stations). Since the number of alternatives is far too high to be expressed in a comprehensible question, the LGM does not have any templates available to express this message. In this situation, i.e., when the DM message cannot even be partly expressed, the LGM currently generates the ‘backchanneling’ utterance *Hm...*, as shown below, and indicates to the DM that the generated utterance has no semantic content by giving a pair of empty brackets as feedback. The idea behind the backchanneling strategy is that the *Hm* will encourage the user to continue his or her turn (cf. Aist and Mostow 1999). This solution is not satisfactory, however, since users do not know what kind of reaction is expected from them in such a situation. The best solution would be for the DM to send an alternative set of dialogue acts to the LGM, after having been notified that the LGM could not carry out its latest instructions. However, the DM of OVIS is currently not capable of doing this.

(10) DM: `user.wants.info.the.route.trajectory.origin ?| {a |`
`atn |`
`... |`
`z1 }`

LGM: *Hm.*
 ()

The above examples illustrate that the LGM works under linguistic and practical constraints that the DM does not anticipate, but which may cause the LGM not to carry out all of the instructions from the DM. This is similar to what happens in human sentence production, where there may be a mismatch between the speaker’s conceptual intentions and the possibilities of formulation (Kempen 1977:260). The occasional occurrence of such mismatches in OVIS makes it important to have feedback from the LGM to the DM: it is crucial that the DM knows exactly which message has been conveyed to the user, in order to react correctly to the user’s subsequent utterance.

5.2.5 Variation

An important difference between OVIS and the other D2S-based systems is the role of variation. In applications such as GoalGetter and DYD, variation is used as a means of keeping users from getting bored when listening to several presentations in succession. Both DYD and GoalGetter are meant to provide not only information, but also some entertainment: in these systems, language generation is used as a means to get information across *in a pleasant way*. Information systems like OVIS, however, have as their primary goal to get information across to the user as quickly and efficiently as possible; not to provide entertainment. In such systems 'gratuitous' variation in the system output is unwanted, as users may try to interpret this kind of variation as meaningful even though it is not. In addition, as noted in Section 2.4.3, the lack of predictability caused by variation may hamper the user's processing of the system's utterances.

In OVIS, there is no arbitrary choice between applicable templates: there is always exactly one template that is most appropriate to express the input message (not counting messages concerning non-related fields like (5)). Neither is there any variation in the filling of the template slots. Fixed expressions are used for all domain entities, since first referring to an entity in one way (e.g., *Vlissingen*) and then another (e.g., *the largest town of Zeeland*) is likely to confuse the user. This does not mean that there is a one-to-one mapping from DM messages to syntactic templates; depending on the linguistic context, another template with a different wording may be chosen. An example of this kind of conditional variation is shown in (11), where the DM tries to verify the value of the destination field twice. The first time, the verification question is expressed in the standard way, by means of an elliptic utterance. When the user's answer is not properly recognised, the DM repeats the question. This time, another template is selected, so that the verification question is expressed by means of a more elaborate, non-elliptic utterance.

- (11) a. DM: `user.wants.info.the_route.trajectory.destination ?+ mas`
LGM: To Maarsssen?
b. User: (...? ...)
c. DM: `user.wants.info.the_route.trajectory.destination ?+ mas`
LGM: I understood that you want to travel to Maarsssen; is this correct?
d. User: Yes.

The aim of the variation between (11)a and c is to clarify the intentions of the system and to encourage the user to provide a different

answer, preferably a simple *Yes* or *No*. This variation has a clear function in the dialogue: if the system had used the same wording twice, the user would probably have reacted in the same way, and the recognition problem would have persisted.

5.2.6 Discussion

In this section we have seen that for use in OVIS, several changes have been made to the way the LGM functions. These changes are due to the changed system context (in particular, the interaction between the DM and the LGM) in combination with the general dialogue constraint that only one, possibly complex, question can be asked within a dialogue turn. Among other things, a new element, the level information, has been added to the syntactic templates in OVIS, and the possible occurrence of ‘mismatches’ between the input from the DM and the coherence constraints of the LGM forces the LGM to provide the DM with continual feedback on the actions it has performed.

The linguistic rules employed by the LGM, however, have *not* been changed for use in OVIS. This was not necessary, because – contrary to what might be expected – language generation in OVIS is actually simpler than in a monologue system such as GoalGetter. As noted in Section 5.2.3, system utterances in OVIS are kept short and simple, system prompts generally consist of one sentence only, and all domain entities are referred to using fixed expressions. In addition, the accentuation of the OVIS prompts is fully predictable, since each template is only applicable in one fixed linguistic context, as discussed in Section 5.2.5. The fact that in a dialogue, the linguistic context consists of utterances from both system and user, does not influence language generation in OVIS at all: the LGM simply ignores the utterances of the user, and because of the limited nature of the dialogue, it can do so without any adverse effects.⁵

However, as dialogue systems become more sophisticated, allowing for more complex conversations, context-sensitivity of the system prompts will become increasingly important. This means that it should be studied how a dialogue context influences linguistic phenomena such as anaphora and accentuation. These issues are addressed in the next section.

⁵In this respect, OVIS is not different from most current spoken dialogue systems, as discussed in Section 5.1.

5.3 Context and information status in dialogues

In the preceding chapters, it has been shown that both accentuation and the generation of referring expressions depend on the *status* of the information being talked about. Information status, in its turn, depends (among other things) on the linguistic context: information that is being introduced into the discourse, and thus is not yet part of the linguistic context, is assumed to be *new* to the hearer, whereas information that has already been mentioned, and thus is part of the linguistic context, may be regarded as *given*, or known to the hearer. New information is generally accented by the speaker, and entities that are referred to for the first time must be distinguished from all other domain entities. Given information, on the other hand, is generally deaccented (except when it is *contrastive* to something in the linguistic context) and entities that are already known to the hearer only need to be distinguished from entities that are equally or more familiar to the hearer.

The above holds in a monologue context, where the linguistic context consists entirely of utterances by one speaker. However, in a dialogue, the linguistic context consists of utterances by two dialogue participants. This raises the question of what happens when the current speaker refers to information that has been added to the linguistic context by the other participant. Do the general rules outlined above apply across speakers, or does information introduced by the other dialogue participant have a different status from that introduced by the current speaker, thus affecting accentuation and referring expression generation? This question is addressed in the current section.

First, it is shown that the straightforward relation between givenness/newness and linguistic context that has so far been assumed, cannot be maintained in a dialogue situation. Information that has been put forward by one dialogue participant cannot be regarded as immediately 'known' by the other. Transferring information from the linguistic context to the hearer's knowledge involves a process that has been termed *grounding*. This process is described in Section 5.3.1. The issues of cross-speaker anaphora and accentuation in dialogue are discussed in Sections 5.3.2 and 5.3.3 respectively, taking the notion of grounding into account. It should be noted that these two sections are of a speculative nature; they do not report on finished research, but rather raise many points that should be further (empirically) investigated. The observations put forward in Sections 5.3.2 and 5.3.3 are all based on dialogues between two human speakers; the relevance of these observations for spoken dialogue systems is discussed in Section 5.3.4.

5.3.1 Grounding

In the discussion of monologue generation in the preceding chapters, a very simple model of knowledge transfer was assumed, where all information that is expressed by the system (i.e., is part of the linguistic context) automatically becomes known to the user. The private knowledge of the system corresponds to the values in the input data structure; by expressing these, they are added to the knowledge of the user, thus becoming part of the mutual knowledge or *common ground*.

As noted by Clark and Schaefer 1989, the assumption that all that is said immediately becomes known to the hearer is a severe simplification. It is very well possible that, for instance, the hearer did not hear or did not understand the speaker's utterance. In these cases, the information that has been expressed by the speaker fails to be added to the hearer's knowledge. In the simple model of knowledge transfer assumed so far, the possible occurrence of such communication problems is not taken into account: a perfect understanding on the part of the hearer is assumed. For the generation of monologues, this idealisation is justifiable, since – due to the 'one-way traffic' nature of a monologue – it is impossible for the system to detect and solve communication problems anyway. In a dialogue, however, the situation is different: here, the dialogue participants react to each other's utterances, signalling whether or not they have understood each other. This signalling is part of a process called *grounding* (Clark and Schaefer 1989, Traum 1994): the interactive process of adding information to the common ground. The grounding process typically proceeds in two phases: a *presentation* phase in which the current speaker sends some message to his conversation partner, and an *acceptance* phase in which the other signals whether the message came across correctly or not. Information introduced by one participant is only added to the common ground if the other has signalled his or her understanding.

Since information that has been expressed by one of the dialogue participants is not necessarily immediately grounded, an intermediate information status is required for information that is part of the linguistic context (which means that it is no longer fully private) but which has not yet been grounded. Between the private knowledge of system and user and the common ground, containing the knowledge they share, there is an intermediate tier containing information which has a mixed status: it has been expressed, but it has not yet been added to the common ground, for instance because the user and the system have different ideas on which information has been expressed. For this reason, the middle tier is divided into two parts: a user part, containing the user's

beliefs concerning the linguistic context, and a system part, containing the system's beliefs concerning the linguistic context. Only after system and user have reached a mutual agreement on what is in the linguistic context, the information from the middle tier can be added to the common ground. Because this information is shared, the common ground is not split up between user and system. Grounding of information may take more than one dialogue turn, which can be illustrated using the following example from the OVIS domain:

- (12) a. User: I'd like to go to Maastricht.
 b. OVIS: To Maarssen?
 c. User: No, to Maastricht.
 d. OVIS: To Maastricht?
 e. User: Yes, that's correct.
 f. OVIS: From where do you want to leave?

In (12)a, the user presents the information that he or she wishes to travel to Maastricht. Since the system is not sure it has recognised the user's utterance correctly, it asks a verification question in (12)b, thereby (*i*) signalling to the user that the information presented in (12)a is not yet grounded (acceptance phase) and (*ii*) presenting what the system assumes the user has said (presentation phase). The correction by the user in (12)c also involves both phases: the user signals that the system's assumption is incorrect (acceptance phase) and presents the correct information once again (presentation phase). This time the destination is correctly recognised by the system, but since the system is still not sure of this, again it asks a verification question. The user reacts with a confirmation in (12)e, signalling that the system's assumption is correct. Finally, the system shows that it has understood the user's confirmation by asking a new question. All in all, it has taken user and system five dialogue turns (from (12)a to (12)f) to ground the information provided by the user in (12)a.

5.3.2 Cross-speaker anaphora

In Chapter 4, an algorithm has been presented for the context-sensitive generation of referring expressions in monologues, based on the assumptions that people prefer anaphoric descriptions containing fewer and more general properties than their antecedents, and that they prefer anaphors to be pronominalised whenever possible. These assumptions were largely⁶ supported by the results of an experiment described in Section 4.6, which showed that the assumptions are also valid when

⁶The experiment indicated that more general properties are only preferred if they do not involve a change in wording.

anaphor and antecedent are separated from each other by a sentence. The current section offers some discussion on the validity of these assumptions in a dialogue context, using the three-tiered model of information status presented above. The focus is on *cross-speaker anaphora* (Francez and Berg 1994), where one speaker anaphorically refers to a discourse entity introduced by another speaker. The reader should keep in mind that the observations put forward in this section are mainly based on constructed examples, which means that further empirical investigation of the issues raised here is required.

The following examples are ‘dialogue versions’ of the first part of example (11) from Chapter 4. In the examples, A and B are having a conversation about a pet show they both visited. Each example illustrates one of the first three hypotheses from Section 4.6.

(13) Hypothesis I: omission of properties

A: Joe bought the large black long-haired sausage dog and the small grey pygmy poodle with the perm wave.

B: That doesn’t surprise me; the sausage dog was a bargain.

A: The poodle was very expensive though.

(14) Hypothesis II: more basic values

A: Joe bought the large black long-haired persian cat and the small grey pygmy poodle with the perm wave.

B: That doesn’t surprise me; the cat was a bargain.

A: The poodle was very expensive though.

(15) Hypothesis III: pronominalisation

A: Joe bought the large black long-haired sausage dog.

B: That doesn’t surprise me; it was a bargain.

In the above examples, B indicates that A’s assertion has been understood and accepted by saying *That doesn’t surprise me*, and presents some new information about the entities, or entity, referred to by A. In doing this, B refers to the entities introduced by A just as if they were introduced by himself: by using fewer properties (13); by using more general properties (14); and by pronominalisation in a case where reference is unambiguous (15). In addition, examples (13) and (14) show that A can use a reduced description (*the poodle*) even though the anaphor is separated from its antecedent by B’s utterance. So, once the information provided by one speaker has been grounded, both speakers can freely refer to it. This is as predicted by the three-tiered model of

information status, where no distinction is made between the sources of the information in the common ground.

In examples (13) through (15), speaker B explicitly indicated that the information provided by A could be grounded, before using a reduced description to refer to the dog mentioned by A. Now let us see what happens if B does not give such a signal of acceptance. As noted by Dekker 1997, Groenendijk et al. 1997 and Poesio 1998, it is quite odd for a speaker to make an assertion about an entity introduced by another speaker, before it has been made clear that both speakers are talking about the same entity. Questions and modal statements, on the other hand, are allowed in all circumstances. This can be illustrated by the examples below (after Groenendijk et al. 1997).

- (16) a. A: Joe bought the large black long-haired sausage dog.
 b. B: ?? It was a bargain.
 c. B: Yeah, it was a bargain.
 d. B: Was it a bargain?
 e. B: Then it must have been a bargain.

B's utterance in (16)b does not seem very appropriate as a reaction to (16)a, whereas the utterances in (16)c-e are perfectly acceptable. The only difference between (16)b and (16)c is that in (16)c, B explicitly indicates having grounded (and thus identified the referent of) A's utterance. Without such an explicit indication, it is only marginally acceptable for B to make an assertion concerning the referent introduced by A.⁷ However, even when B has not been able to identify the referent introduced by A, he can still anaphorically refer to it, given that the reference is embedded in a question or a modal statement. These utterance types are allowed, since they do not require the speaker to have identified the referent (or even to have committed to its existence). In other words, within a question or a modal statement speaker B may refer to an entity which, for all he knows, might not exist.

Summarising, it seems that after speaker B has signalled that he has identified an entity introduced by speaker A, he may refer to this entity in the same way as if he had introduced it himself; however, if B does not give such a signal, then he can only refer to this entity as part of a question or a modal statement. So, although grounding does not seem to influence the form of the referring expressions in cross-speaker anaphora,

⁷As noted by Groenendijk et al. 1997, follow-ups like (16)b are only 'more or less' acceptable in very specific circumstances, for instance if A and B are taking turns in telling a story together. In such a situation, it is obvious that A and B are talking about the same entity, so it is not strictly necessary to explicitly acknowledge this.

it does influence the kind of utterance in which the expressions may be used.

Still, in some dialogue situations the use of reduced anaphoric descriptions across speakers is completely impossible. This is the case when the second speaker has not heard correctly what the first speaker has said, and therefore does not know what is in the linguistic context. In such a situation, in order to reconstruct the linguistic context, the second speaker will either ask the first speaker for clarification (*What did you say?*) or present his assumptions on what the other has said, for the other to confirm or correct. Only after the misrecognition has been cleared up, the grounding process can start. An example is shown in (17) below. Note that here, like in the examples shown above, speaker B first signals that he has identified the referent, before making a statement about it.

- (17) A: Joe bought the large black long-haired sausage dog.
 B: The large black long-eared sausage dog?
 A: No, long-haired.
 B: Ah, yes. It was a bargain.

5.3.3 Accentuation

For accentuation in monologues, it has been assumed that words or phrases are deaccented if they express information that is *given*, i.e., part of the linguistic context, and accented if they express information that is new or contrastive. As explained in Section 5.3.1, in a dialogue the linguistic context can be divided into two parts, consisting of information that has been grounded and information that has not (yet) been grounded. This means that in a dialogue context, the notion of givenness may be interpreted in two ways. In the restrictive interpretation, it includes only information that is part of the common ground, whereas in the broad interpretation, it includes all information that is in the linguistic context (and that is not necessarily grounded).

The dialogue examples given in Section 5.3.2 point towards the latter, broad interpretation of givenness. Since speakers can apparently use a reduced description when referring to entities introduced by another speaker, even before grounding has taken place, it may be expected that deaccentuation is also allowed before grounding. This expectation is based on the fact that salience, which guides the use of reduced descriptions, is actually the degree of givenness of an entity. So, if an entity is sufficiently salient for its reference to be reduced, as in examples (13) through (16), it must be deaccentable as well. In LGM terminology, these examples may be regarded as cases of 'object-givenness' across

speakers: givenness based on previous mention of the same entity, in this case by a different speaker.

The results of a small test described in Cruttenden 1993 indicate that there may also be deaccentuation across speakers due to 'concept-givenness'. As discussed in Section 2.5.2, 'concept-givenness' does not involve co-reference, but words that express similar concepts. Cruttenden's test was based on examples like the following, which were translated in various languages:

(18) A: If you don't hurry up, you'll be late.
B: I don't care if we are late.

(19) A: I make the answer sixteen point one.
B: Well, I make it twenty-six point one.

In example (18), most of Cruttenden's informants deaccented the words (corresponding to) *we* and *late*, for instance saying *I don't CARE if we are late*. The first deaccentuation may be seen as a case of object-givenness (since *we* refers to the same people as the *you* in A's utterance) and the second as a case of concept-givenness. Similarly, in (19) most informants deaccented the words (corresponding to) *point one*.

The above examples suggest that it is possible for a speaker to deaccent words and phrases that were used previously by another speaker. Still, in some situations speakers may choose *not* to deaccent contextually given words; this is called 'reaccenting' in Cruttenden 1993. For instance, in situations where the speaker thinks he has misrecognised what the other has said, and repeats what he thinks he has heard, the repeated information is generally reaccented, even though it may be literally the same as what has been said by the previous speaker. Typically, a speaker only reaccents those words he thinks he may have misrecognised, and deaccents the others. This is illustrated in (20). By using utterance (20)b, B indicates that he thinks he may have misheard the entire description provided by A;⁸ in (20)c, he indicates that he is unsure of the first property only.

(20) a. A: Joe bought the large black long-haired sausage dog.
b. B: Did you say the LARGE BLACK LONG-HAIRED SAUSAGE DOG?
c. B: Did you say the LARGE black long-haired sausage dog?

A similar effect can be observed in situations where the speaker has recognised what the other has said, but indicates his doubt or surprise

⁸The accented adjectives in this example are likely to be followed by minor phrase boundaries, which are not shown here. This also holds for example (21)c.

about parts of it by reaccenting these parts. In (21)b below, B reaccents the word *bought*, thereby indicating that he is surprised by, or doubtful of, the asserted relation between the discourse entities introduced by A (which he has clearly identified). In (21)c, B expresses his surprise or doubt concerning the identity of the animal that Joe has bought. Conversely, in (21)d, B is surprised that it was *Joe* who bought this particular animal.⁹

- (21) a. A: Joe bought the large black long-haired sausage dog.
 b. B: He BOUGHT it?
 c. B: He bought the LARGE BLACK LONG-HAIRED SAUSAGE DOG?
 d. B: {HE / JOE} bought it?

Empirical evidence for this relation between reaccenting and the degree of confidence is given by Grice and Savino 1997 (for Italian) and Swerts et al. 1998 (for Japanese). Grice and Savino 1997 studied a corpus of ‘map task’ dialogues between speakers of Bari Italian. In the map task (see Anderson et al. 1991), both speakers have a map containing various landmarks, and the speakers cannot see each other’s maps. One of them, the instruction giver G, verbally instructs the other, the follower F, to set out a route on his or her map. This task is made more difficult by the fact that there may be discrepancies in the positioning of the landmarks on the two maps. Grice and Savino 1997 found that in their corpus, followers only deaccent repeated material if they are confident that the instructor refers to shared information (i.e., a landmark occurring on both maps). Reaccenting of repeated material occurs if the follower has no confidence in the instructions, for instance because the instructor refers to a landmark that is not on the map. An example of this is given below; here the follower obviously fails to locate the restaurant mentioned by the instructor.

- (22) G: Do you have it, restaurant Anima Mia?
 F: ANIMA MIA?!
 (Grice and Savino 1997:32)

Swerts et al. 1998 studied the prosodic features of repetitive utterances in a corpus of Japanese task-oriented dialogues, where one speaker instructed the other to build a particular figure using a set of blocks. The repetitive utterances were labelled in terms of ‘the degree to which the speaker had integrated the given information into his body of know-

⁹As shown by (21)c and (21)d, the use of non-reduced descriptions is quite acceptable in these cases. Note that B’s reference to Joe in (21)d may be pronominalised, but that the definite description in (21)c cannot be replaced by a pronoun. This may be due to the fact that it is generally awkward to accent the pronoun *it*.

ledge" (Swerts et al. 1998:484). The results of the analysis indicated a clear correspondence between prosodic features and degree of information integration; among other things, non-integrated repeats were found to have a higher than average pitch range.

In short, it appears that deaccenting across speakers is allowed, but that speakers tend to reaccent information they find unexpected or untrustworthy (for instance due to recognition problems). A final reason for reaccenting given information in a dialogue is contrast. Contrastive accenting across speakers has been studied by Krahmer and Swerts in press, who elicited contrastive utterances from the participants in a simple dialogue game. During the game, the participants took turns in instructing each other to pick up one specific figure from a set of coloured geometrical figures, which they could both observe. The games were constructed so that each target description of a figure (e.g., *blue square*) occurred in four different contexts: all new (at the beginning of a game), all contrast (e.g., after the other speaker had mentioned a *red circle*), colour contrast (after *red square*) and form contrast (after *blue circle*). Prosodic analysis of the target descriptions in contrastive contexts revealed that most speakers accented the words expressing contrastive information, and deaccented the other words. So, in an all contrast context, both adjective and noun were generally accented (BLUE SQUARE); when only the colour was contrastive, the adjective was accented (BLUE *square*); and when only the form was contrastive, the noun was accented (*blue* SQUARE). However, there were some exceptions to this. For instance, in an all contrast context, a few speakers accented only the adjective or the noun. It turned out that in these cases, the speakers actually signalled a contrast with their *own* last utterance, ignoring the intervening utterance of the other speaker. The two alternative accentuation patterns in an 'all contrast' context are shown in (23) below.

- (23) A: Blue circle.
 B: Red circle.
 A: BLUE SQUARE. / Blue SQUARE.

5.3.4 Discussion

The examples discussed in this section suggest that both deaccentuation and the use of reduced anaphoric descriptions are permissible across speakers in a dialogue. Both appear to apply to what is in the linguistic context, independent of grounding. However, when in doubt about what is in the linguistic context, for instance due to possible misrecognitions, speakers generally do not use deaccented or reduced descriptions when referring to the other speaker's utterance. The latter observation is

particularly relevant for language generation and accentuation in spoken dialogue systems, where misrecognitions are quite common. Until the system has established beyond doubt what is in the linguistic context (i.e., what the user has said), the system should not use a deaccented or reduced description to refer to information that was provided by the user. After confirmation by the user, the information may be grounded, and then further references to it can be deaccented and reduced. An example from the OVIS domain is shown in (24) below. Here the system first accents *Maastricht* and then, after grounding, refers to it as *there*:

- (24) User: I'd like to go to Maastricht.
 OVIS: To MAASTRICHT?
 User: Yes, that's correct.
 OVIS: WHEN do you want to go there?

In the above example, after confirmation by the user, the system immediately grounds the information about the destination town. However, things do not always go so smoothly as in this example. For instance, the system may have problems recognising the user's reaction to the verification question, in which case the verification question must be repeated, as in example (11). In addition, sometimes the verified information cannot be grounded even though it has been confirmed by the user. This happens, for instance, when the town referred to by the user does not have a train station, which gives rise to an inconsistency (see Section 5.2.2). In this case, there seem to be two possibilities. Either the system could use the full, accented town name to convey a lack of confidence (as in examples (21) and (22)) or it could use a deaccented, possibly reduced description to indicate that the town was successfully identified. Both options are shown below in (25). In OVIS, the first option is chosen, to indicate that the user has provided an unexpected value of the *destination* slot. As a general rule, in OVIS only grounded information is deaccented, even though in the light of the examples discussed so far this restriction may be too strong. For examples like (25), it should be empirically tested which is the best accentuation strategy.

- (25) User: I'd like to go to Uden.
 OVIS: To UDEN?
 User: Yes, that's correct.
 OVIS: UDEN does NOT have a TRAIN STATION. /
 { Uden / that town / it } does NOT have a TRAIN STATION.

Finally, it seems that contrastive accentuation is common across speakers, but that – at least in specific circumstances – some speakers only use contrastive accent in relation to their own utterances. In a

spoken dialogue system, in order to determine whether the system utterances are contrastive to the user's, the language generation module must have a representation of the user's utterance at its disposal. When such a representation is available,¹⁰ the system may determine contrast as discussed in Chapter 3 above. Another, much easier, solution would be to simply ignore contrasts with user's utterances. Although this does mimic the behaviour of some speakers (the 'egocentric' speakers of Krahmer and Swerts 1998 and Krahmer and Swerts in press), it does not seem advisable to follow this approach, since it may create the impression that the system is not paying any attention to what the user is saying. In addition, it is very well possible that the 'egocentric' contrasting behaviour only occurs in very limited conditions, such as the simple instructive dialogues discussed by Krahmer and Swerts.

5.4 Detecting information status

In a spoken dialogue system, identification of entities introduced by the user is often problematic due to the possibility of recognition problems. As pointed out in Section 5.3.2, when the system suspects such a problem, it cannot anaphorically refer to the discourse entity introduced by the user, because it is not sure of what is in the linguistic context. Neither can it make any assertions (i.e., present any travel or ticket information) involving the entities introduced by the user. First, the system has to make sure that it is speaking about the same entities as the user. This is done by presenting to the user the information that has been understood by the system. The user then indicates whether this information has been understood correctly or not, by sending either a positive ('go on') or a negative ('go back') signal. Obviously, it is important for the system to be able to distinguish 'go on' from 'go back' signals, since this information is helpful for determining subsequent system actions. This raises the question whether there is a way to distinguish 'go back' from 'go on' signals.

This section describes two studies that focus on the users' signalling of information status in human-machine interactions. First, Section 5.4.1 briefly describes the corpus that was used in both studies. In the next sections it is investigated which cues people actually use to distinguish positive and negative signals in human-machine interaction. Section 5.4.2 discusses the use of linguistic (non-prosodic) cues, whereas Section 5.4.3 focuses on prosodic cues. The section ends with a discussion.

Sections 5.4.2 and 5.4.3 are based on Krahmer et al. 2000a and Krah-

¹⁰This is not always the case, e.g., in OVIS it is not.

mer et al. 2000b respectively. These publications describe joint work of the author with Emiel Krahmer, Marc Swerts and Mieke Weegels.

5.4.1 Corpus

The stimuli used in both studies described below were taken from a corpus collected by Weegels 1999. The corpus consists of 120 dialogues with two Dutch spoken dialogue systems. Both systems are speaker-independent dialogue systems providing train time table information. One of them, ARISE (Sturm et al. 1999), has been developed within a European project. The other, VIOS, is a fully operational information system in use with the Dutch railways. Both systems prompt the user for unknown slots, such as departure station, arrival station, date, etc., in a series of questions. The two systems differ mainly in length of system utterances (ARISE uses shorter prompts than VIOS), speech output (ARISE uses speech synthesis, whereas VIOS uses a simple form of phrase concatenation) and verification strategy (ARISE only uses explicit verification; VIOS primarily uses implicit verification).¹¹

The corpus was collected during an experiment where twenty subjects were asked to query both systems via telephone on a number of train journeys. They were asked to perform three simple travel queries on each system (in total six tasks). Two similar sets of three queries were constructed, to prevent literal copying of subjects' utterances from the first to the second system. The order of presenting systems and sets was counterbalanced.

5.4.2 Linguistic cues

From studies of human-human conversation it is known that 'go back' signals are comparatively marked (see e.g., Swerts et al. 1998) to make sure that the dialogue partner is immediately made aware of a communication problem. By analogy, it may be expected that when a system utterance makes it clear that something is wrong, users spend more effort on their signals as well, in order to prevent complete breakdown of the communication. Based on the idea that speakers want to finish the dialogue successfully as soon as possible and with minimal effort,

¹¹When performing 'implicit verification', the system mentions the information to be verified while asking for the value of another slot. For instance, in the question *When do you want to travel to Maastricht?* the system attempts to verify the name of the arrival station while at the same time asking a follow-up question concerning the moment. The advantage of this strategy is that, when information has been correctly recognised by the system, it speeds up the conversation; the downside is that when the system made an error, users become confused (see Weegels 1999).

the following distinction between positive and negative cues may be expected:¹²

POSITIVE ('go on')	NEGATIVE ('go back')
short turns	long turns
unmarked word order	marked word order
confirm	disconfirm
answer	no answer
no corrections	corrections
no repetitions	repetitions
new info	no new info

TABLE 5.1: List of linguistic features and their expected settings for positive and negative cues.

In all cases, the positive cues can be seen as the unmarked settings of linguistic features. For instance, the default word order in a sentence is unmarked (thus, no topicalisation or extraposition). Similarly, it is a positive signal to present new information (which may speed up the dialogue), but not to repeat or correct information (which will definitely not lead to a more swift conclusion of the conversation).

In human-machine interaction, the point at which users become aware of a communication problem is generally when the system asks a verification question. This makes reactions to verification questions suitable material for studying the cues people use in positive and negative signals: when the information presented by the system does not correspond with the user's intention, the user will react to the verification question with a negative signal; otherwise, a positive signal is sent.

The corpus contains 487 user reactions to system verification questions, 298 of which are reactions to verification questions that did not contain any problems. These user utterances constitute positive, 'go on' signals: the user indicates that the information recognised by the system can be grounded, and that the system can go on with the dialogue. The remaining 145 utterances are reactions to verification questions that contained one or more problems, such as a misrecognised time, date or station name, or an incorrect default assumption (where the system assumes, for instance, that the user is talking about the departure time,

¹²In Table 5.1, a disconfirmation is listed as a negative cue. It should be noted that this expectation only holds for disconfirmations in reactions to verification questions, which are investigated in the current section. In other contexts, however, a disconfirmation may also function as a positive cue. This is discussed in Section 5.4.3.

while in reality the user has the arrival time in mind). The user's reactions to these problematic system utterances constitute negative, 'go back' signals: the communication problem must be solved before the information exchange can progress.

All 487 user utterances described above have been hand-annotated in terms of the features listed in Table 5.1. Analysis of the annotated utterances confirms the expected association of marked features with negative signals, and of unmarked features with positive signals: users more often employ the 'go back' cues from Table 5.1 when the preceding system utterance contains a problem, whereas the 'go on' cues are used more often in response to unproblematic system utterances. Many of the listed cues have a high informativity. For instance, if the user's answer contains a marked word-order, then it is highly likely that the preceding system utterance contained a problem. The downside is that some of the highly informative cues occur rather infrequently. However, *combinations* of features can compensate for this and thus serve as good indicators of information status. Experiments using memory based learning techniques (with the IB1-GR algorithm, see Aha et al. 1991 and Daelemans et al. 1999) have shown that it is possible to predict in 97% of the cases whether or not the preceding system utterance was problematic on the basis of the user's utterance, by looking at all features. This shows that taking combinations of cues into account provides a reliable indicator of problems. But one has to keep in mind that these experiments were performed with hand-annotated data and that there is a considerable gap between such data and the raw output of a speech recognition engine. It remains an empirical question to what extent the positive and negative signals from Table 5.1 can be recovered automatically. In any case, it is to be expected that shifting the analysis from hand-annotated data to word graphs will decrease the percentage of correctly predicted communication problems. This implies that there is definitely room for improvement. Therefore, one possible extension is to include another set of characteristics of user utterances in the prediction: a number of prosodic features.

5.4.3 Prosodic cues

In parallel to the non-prosodic cues, discussed in Section 5.4.2, it may be assumed that negative signals are also *prosodically* marked compared to positive signals (cf. Swerts and Ostendorf 1997). This expectation is supported by a bulk of work on hyperarticulate speech (e.g., Levow 1998, Oviatt et al. 1998, Soltau and Waibel 1998), a speaking style which is typically adopted in reaction to speech recognition problems. Typically, hyperarticulate speech has – among other things – an increased pitch

and longer duration. Such marked prosodic features also show up in human-human conversations with a relatively higher risk of communication failures (Traunmüller and Eriksson 2000). This leads to the expectations in Table 5.2 regarding prosodic features and the predicted settings for positive and negative signals.

For an adequate comparison of the prosodic features of ‘go on’ and ‘go back’ signals, *one* type of utterance is needed which may serve as a ‘go back’ signal in one context while it serves as a ‘go on’ signal in another context. The corpus contains such an utterance, namely a negative answer to different types of system prompts. To illustrate this, consider the following two questions from the corpus:

- (26) a. Do you want to go from Eindhoven to Swalmen?
 b. Do you want me to repeat the connection?

Both (26)a and (26)b are yes/no questions and to both *No* is a perfectly natural answer. However, the two questions serve a rather different goal. Question (26)a is an (explicit) attempt of the system to verify some pieces of information that it has recently gathered (the departure and arrival station). If the user responds to this question with a disconfirmation, this is definitely a ‘go back’ signal: the user indicates that at least one of the system’s beliefs is incorrect. Question (26)b, on the other hand, is not an attempt of the system to verify its beliefs, and hence it cannot represent incorrect system beliefs. A subsequent *No* from the user thus serves as a ‘go on’ signal. The two types of disconfirmations, being lexically similar but functionally different, constitute minimal pairs from a dialogue perspective, which makes it possible to check whether the various occurrences of this utterance vary prosodically as a function of their context. In this way, they form ideal, naturally occurring, speech materials for investigating the role of prosody in information grounding.

For prosodic analysis, a random selection was made of 109 negative

Features	POSITIVE (‘go on’)	NEGATIVE (‘go back’)
Boundary tone	low	high
Duration	short	long
Pause	short	long
Delay	short	long
Pitch range	low	high

TABLE 5.2: List of prosodic features and their expected settings for positive and negative cues

answers to yes/no questions from the corpus (7 speakers). Of these, 41 are answers to system questions that are not verification questions, such as (26)b above. These disconfirmations constitute positive signals. The remaining 68 answers are reactions to verification questions containing a problem. These disconfirmations constitute negative signals. It should be noted here that not all disconfirmations consist of the word *no* only; in most cases, the *no* is followed by other lexical material. Disconfirmations following a problematic verification question are often not preceded by *no*, in which case they consist of corrected material only.

Acoustic analysis of the 109 disconfirmations in terms of the prosodic features from Table 5.2 reveals that, as expected, negative signals are comparatively marked with respect to prosody. First, negative signals differ from positive ones in that the word *no* —when it occurs— in these utterances is longer and generally has a high boundary tone. Second, compared to positive signals, there is a longer delay after a problematic system prompt before users respond. Third, in utterances where the word *no* is followed by other words, the interval between *no* and the remainder of the utterance is longer following a problematic system utterance than following an unproblematic one. Fourth, after a problematic yes/no question, the remainder of the answer usually contains a high-pitched accent to mark corrected information, whereas in the unproblematic case this part of the utterance is usually prosodically unmarked. Finally, in reaction to a problem, the part after *no* tends to be longer in number of words, which is in agreement with the general finding discussed in Section 5.4.2. A perception experiment described in Krahmer et al. 2000a shows that human hearers can indeed use some of these prosodic features to distinguish positive from negative cues, without having access to context information.

These findings can easily be related to the respective functions of the two kinds of disconfirmation. A ‘go on’ disconfirmation is simply an answer to the question and does not address any underlying assumptions of the system. In principle, a single *no* is a sufficient answer; any other words in the reaction are politeness phrases such as *thank you*, which follow more or less automatically and provide no further information. This explains their lack of accents as well as the short pauses between the *no* and the rest of the answer. If a yes/no question from the system contains a problem, however, just answering *No* might be sufficient but is not very cooperative. Assuming that the user wants the dialogue to be over as soon as possible, it is more efficient to immediately *correct* the system. Since the user has to reconstruct where the system’s assumptions are incorrect, and has to formulate an adequate reaction for that particular context, it may be assumed that the cognitive load for

the user is relatively high. As Levelt 1989 argues, there is a close correspondence between the cognitive load and the length of pauses, which might explain why both the delay in answering and the pause between the *no* and the rest of the answer are longer in the case of problems.

5.4.4 Discussion

The findings presented in this section provide potentially useful information for spoken dialogue systems which monitor information status: if a question is followed by a user utterance which has various marked properties (such as relatively many words, corrections, long delays, accented words), the system can be fairly certain that the information it tried to verify is not in agreement with the user's intentions. If, on the other hand, the user utterance does not contain such features, then it is highly likely that the verified information is correct. Using a systematic and reliable strategy to decide whether or not there are communication problems may be very useful in a number of situations.

For language generation, it is essential to know whether information has been understood correctly. As seen in Section 5.3, if this is not the case, it is impossible to refer to this information using a reduced or deaccented anaphoric description. In addition, no assertions can be made concerning this information. Detecting information status based on user cues may also be useful for several system decisions which are not directly related to language generation. For instance, after detection of a communication problem the system may decide to switch to a different speech recognition engine. Levow 1998 reports that in interactions with the Sun Microsystems SpeechActs system (Yankelovich et al. 1995), the probability of experiencing a recognition error after a correct recognition is 16%, but immediately after an incorrect recognition it is 44%. This increase is probably caused by the fact that speakers use hyperarticulate speech when they notice that the system had a problem recognising their previous utterance, so it might be beneficial to switch to a speech recogniser trained on hyperarticulate speech if there are communication problems (cf. Oviatt et al. 1998, Hirschberg et al. 1999). In addition, for a system using an implicit verification strategy (see footnote 11), it might be useful to change to explicit verification when communication problems arise.

5.5 Summary

In this chapter, language generation in spoken dialogue systems has been discussed, starting from natural language generation in current applications and then moving on to some issues that are expected to become relevant for language generation in future dialogue systems.

First, it has been discussed how the LGM of D2S can be used for the generation of system utterances in a spoken dialogue system called OVIS. Several adaptations had to be made to the LGM, due to its interaction with the system's dialogue management module, in combination with the restriction that in a dialogue, the system's utterances should 'fit' within one dialogue turn. At the same time, system utterances in OVIS are fairly short and simple, and do not show much variation. As a consequence, referring expression generation and accent assignment can be performed in a simple fashion in OVIS, without taking the user's utterances into account.

It may be assumed that, as dialogue systems become more sophisticated, this approach will no longer be sufficient and that context-sensitivity of the system prompts will become increasingly important. Therefore, a preliminary, informal investigation has been carried out into the influence of dialogue context on information status. One of the observations that have been put forward is that a dialogue system can only make assertions concerning information that has been grounded, i.e., accepted by both system and user. Before grounding can take place, user and system must ascertain that they are speaking about the same thing. In a dialogue system, this may be problematic due to frequently occurring communication problems, which are caused primarily by speech recognition errors.

A dialogue system largely depends on signals from the user to establish whether or not there has been a communication problem. Analysis of a dialogue corpus in terms of several linguistic and prosodic features has revealed that in the case of communication problems, speakers put much more linguistic and prosodic effort into their reaction. Several features have been located that may help the system to determine automatically whether the user sent a positive signal, indicating that the information presented by the system is correct and may be grounded, or a negative signal, which means there is some communication problem.

Conclusion

This thesis deals with language generation in context, where the word *context* is used in three different senses. First, it has been investigated how information from the *linguistic context* can be taken into account during language generation, in order to achieve natural and coherent output. Second, this issue has been examined for language generation in both a *monologue* and a *dialogue* context. Third, the thesis deals with language generation in the context of a *data-to-speech* system, which combines language and speech generation.

The thesis focuses on the relevance of linguistic context for two of the tasks involved in language generation for data-to-speech: the assignment of contrastive accent and the generation of referring expressions. In Chapters 3 to 5, it is discussed which aspects of the linguistic context are relevant for the above-mentioned tasks and how these aspects can be taken into account so as to produce natural and coherent output. These issues are thoroughly discussed from the point of view of monologue generation, and subsequently they are reconsidered from a dialogue perspective.

Here, a general overview is given of the work presented in this thesis, followed by some pointers to future work.

6.1 General overview

Chapter 1 sketches the general background for the research presented in this thesis, providing a brief overview of current approaches to language and speech generation. The standard tasks for text generation are discussed, and it is argued that in data-to-speech generation, the language generation component should also be responsible for the task of prosody computation. This way, linguistic information that is already available in the language generation component can be used for the assignment

of prosodic markers, without having to obtain the relevant information through analysis of the generated text.

Chapter 2 introduces D2S, the data-to-speech system which served as a framework for the research described here. The workings of D2S are explained using examples from GoalGetter, a system which has been developed as part of this research. In particular, the language generation module of D2S is discussed in detail, including a formal presentation of its main generation algorithm. The most distinguishing characteristic of D2S is that language generation is done using so-called syntactic templates: syntactic tree structures with open slots for variable information. The use of syntactic templates, which are handmade and domain-specific, greatly simplifies the task of sentence realisation in D2S. Other tasks, such as the computation of prosody and the generation of anaphoric expressions, are carried out in a more sophisticated fashion, making use of a representation of the linguistic context which records – among other things – which discourse entities have been mentioned, and how and when they were referred to. The purposes for which this contextual information is used include the checking of accessible antecedents for anaphors and the deaccentuation of words that express previously mentioned information. Most of the research results presented in this thesis have been implemented in applications of the D2S system, which proved to be a suitable framework for a wide range of applications with different domains, languages and interaction styles.

To achieve natural prosody, it is necessary to determine which words in the generated output express contrastive information, since these words are generally accented, even if they have been previously mentioned. In Chapter 3, it is discussed how information from the linguistic context can be used to take contrast into account during prosody computation. In most previous approaches to the determination of contrastive information, contrast is related to some form of parallelism between consecutive sentences. In a small perception experiment, this relation has been confirmed. The results of the experiment suggest that people use contrastive accent to emphasise the points of difference between similar events. To model this behaviour in a language generation system, it is necessary to record not only which entities have been mentioned in the linguistic context, but also which events they have been engaged in, and which role they played in these events. Based on this idea, a practical method has been developed for contrast detection, making use of the data structures used for language generation in D2S. If two consecutive sentences express the same type of data structure, they express the same kind of event and are therefore considered as potentially contrastive. An accent is then assigned to the phrases that express the different parts of

the data structures. Unlike in previous approaches, there is no restriction on the kind of entities that may be considered contrastive; this is in line with the observation that, given the ‘right’ context, any two items may stand in contrast to each other. Another advantage of the proposed approach is that it allows for the automatic detection of contrasts between sentences that describe similar events using different surface structures. In other approaches, detecting such contrasts requires the explicit encoding of world knowledge.

Chapter 4 focuses on another important task in language generation: the generation of referring expressions. When describing a domain entity, the generation system must choose which of its properties to express. This choice depends not only on the properties of the other entities in the domain, but also on the salience of the intended referent. Salience, in its turn, depends – among other things – on the linguistic context: an entity which has been recently mentioned is likely to be more salient than an entity which has not. Based on this idea, an existing, well-known algorithm for the generation of definite descriptions has been adapted so that it takes the linguistic context into account when choosing the properties that are to be included in a description. Only those properties are selected which help to distinguish the intended referent from other entities with the same or a higher salience weight, and this allows for the generation of reduced anaphoric descriptions. Generally, a full description of an entity is only used the first time the entity is mentioned. As the entity becomes more salient in comparison to the other entities in the domain, fewer properties are needed to describe it. This way, coherence and natural variation are achieved, while avoiding the continuous repetition of full descriptions. For modelling salience, a method is proposed which is a synthesis of two existing approaches to salience determination. In addition, an experiment is described which shows that people generally prefer the use of reduced anaphoric descriptions, and it is discussed how the proposed algorithm may be further adapted for the generation of other types of expressions, such as pronouns and relational descriptions.

Finally, in Chapter 5 it is argued that for language generation in a dialogue, the informal definition of linguistic context given in Chapter 1 (“what has been said previously in the text being generated”) is too limited. In a dialogue, the linguistic context does not only consist of the generated system utterances, but also of the user’s contributions to the dialogue. These user contributions provide valuable information on the status of the information being exchanged; this information can only be regarded as mutually known after both user and system have indicated their acceptance. Especially in human-machine dialogues, un-

derstanding may be hampered by poor recognition or incorrect default assumptions on the part of the system, and as long as it is not clear to both parties what is in the linguistic context, deaccentuation or anaphoric reference are impossible. In addition, the system should not make any assertions about information provided by the user before any possible misunderstandings have been cleared up. Automatic detection of communication problems by the system may be helped by the findings from a corpus study revealing that user's reactions to communication problems have comparatively marked linguistic and prosodic features. Apparently, people put more effort into these utterances in order to make sure that the dialogue partner is made aware of the problem.

6.2 Future work

The work presented in the various chapters of this thesis points towards several issues that require further research. An overview of these is given here.

Some improvements to the language generation method used in D2S, discussed in Chapter 2, are proposed in this thesis. Several other improvements are still possible. A first step might be to move towards a more grammar-oriented approach to sentence realisation, and to generalise the local conditions that are used for text 'planning'. This would result in more general syntactic templates, which might be reused across different applications. Another point for improvement is the frequent use of random choice within the generation algorithm. In order to achieve variation within and between the generated texts, random choices are made between 'equally appropriate' topics, templates, word orders, and words. Two remarks are in order here: first, making a random choice is not a guarantee for optimal variation, and second, sometimes the different possibilities are not equally appropriate. Therefore, in situations where variation is really important, a more sophisticated method may be required to achieve it, whereas at other points, choices should be made on linguistic grounds, not on a random basis.

The method proposed for contrast detection in Chapter 3 is still incomplete, in that it does not provide a precise answer to the question which events are similar, and thus potentially contrastive, and which are not. It would be interesting to investigate the suitability of different existing ontologies for the contrast detection method proposed here, and to test the method in various domains. In addition, experimental evidence is still required for the assumption that people have a preference for contrastive accent in cases of 'implicit contrast', where two similar events are described in different ways. (In Chapter 3 such a pref-

erence was only proven for events being described using the same verb and the same syntactic construction.) Another point for future research is the investigation of (possible) contrasts between sentences that are not contiguous. Finally, the apparent relation between coherence and accentuation, discussed in Section 3.5.6, should be further investigated.

In Chapter 4 it is argued that salience of discourse entities plays a crucial role in referring expression generation, and two existing methods for determining linguistic salience are briefly discussed and compared. A further, structural investigation into the suitability of different methods for salience determination for use in referring expression generation is still required. It would also be interesting to take other than linguistic sources of salience into account, such as ‘inherent’ salience of specific entities, situational and spatial salience. Some other issues which have not been dealt with in this chapter are the treatment of relative properties (which, for the sake of simplicity, have been treated as if they were absolute), and the generation of plurals and indefinites. These issues should be addressed in future research. Finally, the generation of relational descriptions requires further empirical research, in particular concerning the ordering of relations and the subsumption hierarchy of types of spatial relations.

As discussed in Chapter 5, the syntactic realisation of short dialogue prompts is a rather limited task when compared with full monologue generation. For the former task, a grammar-based approach may be more suitable than the language generation method used in D2S. However, the D2S method may be appropriate for language generation in more complex dialogues, with longer, less predictable utterances. For such applications, the Discourse Model will have to be extended so that user utterances can be represented and taken into account for the generation of referring expressions and accent assignment. It should be noted that most of the observations concerning cross-speaker anaphora and deaccentuation in dialogue are based on constructed examples and intuitive judgements. In order to make stronger claims concerning these issues, further empirical research is required. Another important point for future research suggested by this chapter, is the question whether the features marking positive or negative user signals, discussed in Sections 5.4.2 and 5.4.3, can be detected automatically in a spoken dialogue system.

6.3 Final word

Language generation involves many different, challenging tasks, only a few of which could be discussed in this thesis. Likewise, ‘context’ is

a nearly all-embracing notion, and this research has necessarily been limited to only those aspects of linguistic context that were found to be relevant for the language generation tasks being investigated. This is not to say that there are no other contextual aspects that may be relevant for these tasks; also, context should doubtlessly be taken into account in all generation tasks, not only those discussed here.

The current work has shown that taking context into account is beneficial for various language generation tasks. It will be interesting to see whether other generation tasks will likewise profit from paying closer attention to context.

Bibliography

- Aha, D., D. Kibler, and M. Albert. 1991. Instance-based learning algorithms. *Machine Learning* 6:37–66.
- Aist, G., and J. Mostow. 1999. Measuring the effects of backchanneling in computerized oral reading tutoring. In *Proceedings of the ESCA International Workshop on Dialogue and Prosody*, 175–180.
- Akmajian, A., and R. Jackendoff. 1970. Coreferentiality and stress. *Linguistic Inquiry* 1(1):124–126.
- Alshawi, H. 1987. *Memory and Context for Language Interpretation*. Cambridge: Cambridge University Press.
- Alshawi, H., and R. Crouch. 1992. Monotonic semantic interpretation. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL'92)*, 32–39. Newark, USA.
- Anderson, A., M. Bader, E. Bard, E. Boyle, G.M. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo, H. Thompson, and R. Weinert. 1991. The HCRC Map Task corpus. *Language and Speech* 34:351–366.
- André, E., G. Herzog, and T. Rist. 1988. On the simultaneous interpretation of real world image sequences and their natural language description: the system SOCCER. In *Proceedings of the 8th European Conference on Artificial Intelligence (ECAI'88)*, 449–454. München, Germany.
- Aust, H., M. Oerder, F. Seide, and V. Steinbiss. 1995. The Philips automatic train timetable information system. *Speech Communication* 17:249–262.
- Baart, J.L.G. 1987. *Focus, Syntax and Accent Placement*. Doctoral dissertation, University of Leiden.
- Bateman, J. 1992. The theoretical status of ontologies in natural language processing. In *Proceedings of the Workshop on Text Representation and Domain Modelling – Ideas from Linguistics and AI*, ed. S. Preuss and B. Schmitz, 50–99. Berlin, Germany. KIT-Report 97.
- Bateman, J., and R. Henschel. 1999. From full generation to ‘near-templates’ without losing generality. In *Proceedings of the KI'99 Workshop ‘May I Speak Freely?’*, 13–18. Saarbrücken, Germany. DFKI.

- Bateman, J., and E. Teich. 1995. Selective information presentation in an integrated publication system: An application of genre-driven text generation. *Information Processing and Management* 31(5):753–767.
- Beun, R.-J., and A. Cremers. 1998. Object reference in a shared domain of conversation. *Pragmatics & Cognition* 6(1/2):121–152.
- Bierwisch, M. 1983. Semantische und konzeptuelle Repräsentation lexikalischer Einheiten. In *Untersuchungen zur Semantik*, ed. R. Ruzicka and W. Motsch. *Studia grammatica*, Vol. 22, 61–99. Berlin: Akademie-Verlag.
- Birner, B. 1994. Information status and word order: An analysis of English inversion. *Language* 70:233–259.
- Blomberg, M., R. Carlson, K. Elenius, B. Granström, K. Gustafson, S. Hunnicutt, R. Lindell, and L. Neovius. 1993. An experimental dialogue system: Waxholm. In *Proceedings of the 3rd European Conference on Speech Communication and Technology (Eurospeech'93)*, 1867–1870. Madrid, Spain.
- Bock, J., and J. Mazzella. 1983. Intonational marking of given and new information. *Memory and Cognition* 11(1):64–76.
- Bod, R. 1998. Spoken dialogue interpretation with the DOP model. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL'98)*, 138–144. Montreal, Canada.
- Bolinger, D. 1986. *Intonation and its Parts: Melody in Spoken English*. London: Arnold.
- Bonnema, R., R. Scha, and R. Bod. 1997. A DOP model for semantic interpretation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97)*, 159–167. Madrid, Spain.
- Bosch, A. van den. 1997. *Learning to Pronounce Written Words. A Study in Inductive Language Learning*. Doctoral dissertation, Maastricht University.
- Brown, G. 1983. Prosodic structure and the given/new distinction. In *Prosody: Models and Measurements*, ed. D. R. Ladd and A. Cutler. 67–77. Berlin: Springer Verlag.
- Busemann, S., and H. Horacek. 1998. A flexible shallow approach to text generation. In *Proceedings of the 9th International Workshop on Natural Language Generation (IWNLG'98)*, 238–247. Niagara-on-the-Lake, Canada.
- Cahill, L., C. Doran, R. Evans, C. Mellish, D. Paiva, M. Reape, D. Scott, and N. Tipper. 1999. In search of a reference architecture for NLG systems. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99)*. KIT Report 97. Toulouse, France.
- Carenini, G., V.O. Mittal, and J.D. Moore. 1994. Generating patient-specific interactive natural language explanations. In *Proceedings of the 18th An-*

- nual Symposium on Computer Applications in Medical Care (SCAMC'94)*. Washington D.C., USA.
- Chafe, W.L. 1976. Givenness, contrastiveness, definiteness, subjects, topics and points of view. In *Subject and Topic*, ed. C. N. Li. 25–55. New York: Academic Press.
- Chomsky, N. 1971. Deep structure, surface structure, and semantic interpretation. In *Semantics: An Interdisciplinary Reader in Philosophy, Linguistics and Psychology*, ed. D. Steinberg and L. Jakobovits. Chap. 183–216. Cambridge: Cambridge University Press.
- Chomsky, N. 1981. *Lectures on Government and Binding*. Dordrecht: Foris.
- Claassen, W. 1998. Improving the usability of spoken presentations of telephone information services. Deliverable 73. Priority Programme on Language and Speech Technology, NWO, The Netherlands.
- Claassen, W. 1999. Using speech shadowing and sentence-by-sentence listening to assess the usability of synthetic speech presentations: Two pilot studies. Deliverable 80. Priority Programme on Language and Speech Technology, NWO, The Netherlands.
- Claassen, W. 2000. Using recall measurements and subjective ratings to compare the usability of railroad travel plans presented by telephone: two experiments. Deliverable 123. Priority Programme on Language and Speech Technology, NWO, The Netherlands.
- Clark, H. 1975. Bridging. In *Theoretical Issues in Natural Language Processing*, ed. R. Schank and B. Nash-Webber. 188–193. Cambridge: MIT Press.
- Clark, H., and E. Schaefer. 1989. Contributing to discourse. *Cognitive Science* 13:259–294.
- Clark, H., and D. Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition* 22:1–39.
- Coch, J. 1996. Evaluating and comparing three text-production techniques. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, 249–254.
- Cole, R., L. Hirschman, L. Atlas, M. Beckman, A. Biermann, M. Bush, M. Clements, J. Cohen, O. Garcia, B. Hanson, H. Hermansky, S. Levinson, K. McKeown, N. Morgan, D. Novick, , M. Ostendorf, S. Oviatt, P. Price, H. Silverman, J. Spitz, A. Waibel, C. Weinstein, S. Zahorian, and V. Zue. 1995a. The challenge of spoken language systems: Research directions for the nineties. *IEEE Transactions on Speech and Audio Processing* 3(1):1–21.
- Cole, R., J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue (ed.). 1995b. *Survey of the State of the Art in Human Language Technology*. Cambridge: Cambridge University Press.
- Cote, S. 1998. Ranking forward-looking centers. In *Centering Theory in Discourse*, ed. M. Walker, A. Joshi, and E. Prince. 55–69. New York: Oxford University Press.

- Cruse, D.A. 1977. The pragmatics of lexical specificity. *Journal of Linguistics* 13:153–368.
- Cruttenden, A. 1986. *Intonation*. Cambridge: Cambridge University Press.
- Cruttenden, A. 1993. The deaccenting and reaccenting of repeated lexical items. In *ESCA Workshop on Prosody*. Working Papers, Vol. 41. Department of Linguistics and Phonetics, Lund, Sweden.
- Daelemans, W., J. Zavrel, K. van der Sloot, and A. van den Bosch. 1999. TiMBL: Tilburg Memory Based Learner, version 2.0, reference guide. Technical Report 99-01. ILK, Tilburg, The Netherlands.
- Dale, R. 1992. *Generating Referring Expressions: Constructing Descriptions in a Domain of Objects and Processes*. Cambridge: MIT Press.
- Dale, R., and N. Haddock. 1991. Generating referring expressions involving relations. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, 161–166. Berlin, Germany.
- Dale, R., and E. Reiter. 1995. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science* 18:233–263.
- Deemter, K. van. 1994a. Contrastive stress, contrariety, and focus. In *Focus & Natural Language Processing, Volume 1: Intonation and Syntax*, ed. P. Bosch and R. van der Sandt. 39–49. Working Papers of the Institute for Logic & Linguistics., No. 6. Cambridge: Cambridge University Press.
- Deemter, K. van. 1994b. What's new? A semantic perspective on sentence accent. *Journal of Semantics* 11:1–31.
- Deemter, K. van. 1998. A blackboard model of accenting. *Computer Speech and Language* 12:143–164.
- Deemter, K. van. 1999. Contrastive stress, contrariety, and focus. In *Focus: Linguistic, Cognitive, and Computational Perspectives*, ed. P. Bosch and R. van der Sandt. 3–17. Cambridge: Cambridge University Press.
- Deemter, K. van, E. Kraemer, and M. Theune. 1999. Plan-based vs. template-based NLG: A false opposition? In *Proceedings of the KI'99 Workshop 'May I Speak Freely?'*, 1–5. Saarbrücken, Germany. DFKI.
- Deemter, K. van, J. Landsbergen, R. Leermakers, and J. Odijk. 1994. Generation of spoken monologues by means of templates. In *Proceedings of the 8th Twente Workshop on Language Technology (TWLT 8): Speech and Language Engineering*, 87–96. Enschede, The Netherlands.
- Deemter, K. van, and J. Odijk. 1997. Context modeling and the generation of spoken discourse. *Speech Communication* 21(1/2):101–121.
- Deemter, K. van, G. van der Hoeven, R. Leermakers, J. Odijk, and F. Uittenbogaard. 1996. The use of natural language in a browsing interface. IPO MS 1142.
- Dekker, P. 1997. On first order information exchange. In *Proceedings of Mundial 97, the München Workshop on the Formal Semantics and Pragmatics of Dialogue*, ed. A. Benz and G. Jäger. CIS München, Germany.

- Delin, J., and R. Zacharski. 1997. Pragmatic determinants of intonation contours for dialogue systems. *International Journal of Speech Technology* 1:109–120.
- Dirksen, A. 1992. Accenting and deaccenting: A declarative approach. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING'92)*, 865–869. Nantes, France.
- Dirksen, A., and H. Quené. 1993. Prosodic analysis: The next generation. In *Analysis and Synthesis of Speech: Strategic Research Towards High-Quality Text-to-Speech Generation*, ed. V. van Heuven and L. Pols. 131–144. Berlin - New York: Mouton de Gruyter.
- Donzel, M. van. 1999. *Prosodic Aspects of Information Structure in Discourse*. Doctoral dissertation, University of Amsterdam.
- Fellbaum, C. (ed.). 1998. *WordNet: An Electronic Lexical Database*. Cambridge: MIT Press.
- Francez, N., and J. Berg. 1994. A multi-agent extension of DRT. In *Proceedings of the 1st International Workshop on Computational Semantics*, ed. H. Bunt, R. Muskens, and G. Rentier, 81–90. Tilburg, The Netherlands.
- Gardent, C. 1997. Parallelism, HOU and deaccenting. CLAUS-report 85. Universität des Saarlandes, Saarbrücken, Germany.
- Gardent, C. 2000. Deaccenting and Higher-Order Unification. *Journal of Logic, Language, and Information* 9:313–338.
- Gardent, C., and M. Kohlhase. 1997. Computing parallelism in discourse. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, 1016–1021. Tokyo, Japan.
- Gardent, C., M. Kohlhase, and K. Konrad. 1998. Higher-order coloured unification: A linguistic application. CLAUS-report 101. Universität des Saarlandes, Saarbrücken.
- Geldof, S., and W. van de Velde. 1997. An architecture for template based (hyper)text generation. In *Proceedings of the 6th European Workshop on Natural Language Generation (EWNLG'97)*, 28–37. Leiden, The Netherlands.
- Geurts, B. 1995. *Presupposing*. Doctoral dissertation, University of Osnabrück.
- Gigi, E., and L. Vogten. 1997. A mixed-excitation vocoder based on exact analysis of harmonic components. *IPO Annual Progress Report* 32:105–110.
- Gordon, P., B. Grosz, and L. Gilliom. 1993. Pronouns, names, and the centering of attention in discourse. *Cognitive Science* 17:311–347.
- Grice, H.P. 1975. Logic and conversation. In *Syntax and Semantics: Vol. 3, Speech Acts*, ed. P. Cole and J. Morgan. 43–58. New York: Academic Press.
- Grice, M., and M. Savino. 1997. Can pitch accent type convey information status in yes-no questions? In *Proceedings of the Workshop on Concept-to-Speech Generation Systems, ACL/EACL'97*, 29–33. Madrid, Spain.

- Groenendijk, J., M. Stokhof, and F. Veltman. 1997. Coreference and modality in the context of multi-speaker discourse. In *Context Dependence in the Analysis of Linguistic Meaning*, ed. H. Kamp and B. Partee, 195–216. IMS Stuttgart, Germany.
- Grosz, B.J., A.K. Joshi, and S. Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics* 21(2):203–225.
- Grosz, B.J., and C.L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational Linguistics* 12(3):175–204.
- Hajičová, E. 1993. Issues of sentence structure and discourse patterns. *Theoretical and Computational Linguistics* 2.
- Halliday, M.A.K. 1967. Notes on transitivity and theme in English. *Journal of Linguistics* 3:199–244.
- Hart, J. 't, R. Collier, and A. Cohen. 1990. *A Perceptual Study of Intonation: An Experimental Phonetic Approach to Speech Melody*. Cambridge: Cambridge University Press.
- Hirschberg, J. 1992. Using discourse context to guide pitch accent decisions in synthetic speech. In *Talking Machines: Theories, Models and Designs*, ed. G. Bailly, C. Benoit, and T.R. Sawallis. 367–376. Amsterdam: Elsevier Science Publishers B.V.
- Hirschberg, J., D. Litman, and M. Swerts. 1999. Prosodic cues to recognition errors. In *Proceedings of the International Workshop on Speech Recognition and Understanding (ASRU'99)*. Keystone, USA.
- Hiyakumoto, L., S. Prevost, and J. Cassell. 1997. Semantic and discourse information for text-to-speech intonation. In *Proceedings of the Workshop on Concept-to-Speech Generation Systems, ACL/EACL'97*, 47–56. Madrid, Spain.
- Hobbs, J.R., and A. Kehler. 1997. A theory of parallelism and the case of VP ellipsis. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97)*, 394–401. Madrid, Spain.
- Hoeven, G. van der, J. Andernach, S. van de Burgt, G.-J. Kruijff, A. Nijholt, J. Schaake, and F. de Jong. 1996. SCHISMA: A natural language accessible theatre information and booking system. In *Proceedings of the 1st International Workshop on Applications of Natural Language to Databases (NLDB'95)*, 271–285. Versailles, France.
- Horacek, H. 1995. More on generating referring expressions. In *Proceedings of the 5th European Workshop on Natural Language Generation (EWNLG'95)*, 43–85. Leiden, The Netherlands.
- Horacek, H. 1997. An algorithm for generating referential descriptions with flexible interfaces. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of*

- the European Chapter of the Association for Computational Linguistics (ACL/EACL'97)*, 206–213. Madrid, Spain.
- Hudson-D'Zmura, S., and M. Tanenhaus. 1998. Assigning antecedents to ambiguous pronouns: The role of the center of attention as the default assignment. In *Centering Theory in Discourse*, ed. M. Walker, A. Joshi, and E. Prince. 199–226. New York: Oxford University Press.
- Jelinek, F. 1997. *Statistical Methods for Speech Recognition*. London: MIT Press.
- Joshi, A. 1987. An introduction to Tree Adjoining Grammars. In *Mathematics of Language*, ed. A. Manaster-Ramer. 87–114. Amsterdam: John Benjamins.
- Kameyama, M. 1986. A property-sharing constraint in Centering. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics (ACL'86)*, 200–206. New York, USA.
- Kameyama, M. 1996. Infeasible semantics and defeasible pragmatics. In *Quantifiers, Deduction and Context*, ed. M. Kanazawa, C. Piñon, and H. de Swart. 111–138. Stanford: CSLI Publications.
- Kamp, H. 1996. Presupposition, focus, anaphora, and ellipsis: What do they have in common and what sets them apart. Handout of a lecture presented on 20 November 1996 at the Max Planck Institute, Nijmegen, The Netherlands.
- Kempen, G. 1977. Conceptualizing and formulating in sentence production. In *Sentence Production: Developments in Research and Theory*, ed. S. Rosenberg. 259–274. Hillsdale: Lawrence Erlbaum Associates.
- Klabbers, E. 1997. High-quality speech output generation through advanced phrase concatenation. In *Proceedings of the COST Workshop on Speech Technology in the Public Telephone Network: Where are We Today?*, 85–88. Rhodes, Greece.
- Klabbers, E. 2000. *Segmental and Prosodic Improvements to Speech Generation*. Doctoral dissertation, Eindhoven University of Technology.
- Klabbers, E., E. Krahmer, and M. Theune. 1998. A generic algorithm for generating spoken monologues. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP'98)*, 2759–2762. Sydney, Australia.
- Klabbers, E., J. Odijk, J.R. de Pijper, and M. Theune. 1996. GoalGetter: From Teletext to speech. *IPO Annual Progress Report* 31:61–75.
- Knuth, D. 1986. *Computers and Typesetting*. Reading: Addison-Wesley Publishing Co.
- Kohlhase, M. 1994. *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle*. Doctoral dissertation, Universität des Saarlandes.
- Krahmer, E. 1998. *Presupposition and Anaphora*. Stanford: CSLI Publications.

- Krahmer, E., J. Landsbergen, and J. Odijk. 1997. A guided tour through LGM; how to generate spoken route descriptions? Report 1182. IPO, Eindhoven, The Netherlands.
- Krahmer, E., and M. Swerts. 1998. Reconciling two competing views on contrastiveness. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP'98)*, 1299–1302. Sydney, Australia.
- Krahmer, E., and M. Swerts. in press. On the alleged existence of contrastive accents. To appear in *Speech Communication*.
- Krahmer, E., M. Swerts, M. Theune, and M. Weegels. 2000a. The dual of denial: Two uses of disconfirmations in dialogue and their prosodic correlates. To appear in *Speech Communication*.
- Krahmer, E., M. Swerts, M. Theune, and M. Weegels. 2000b. Error detection in human-machine interaction. Submitted to *Speech Technology*.
- Krahmer, E., and M. Theune. 1998. Context sensitive generation of referring expressions. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP'98)*, 1151–1154. Sydney, Australia.
- Krahmer, E., and M. Theune. 1999. Efficient generation of descriptions in context. In *Proceedings of the Workshop on the Generation of Nominals, ESSLLI'99*, ed. R. Kibble and K. van Deemter. Utrecht, The Netherlands.
- Krahmer, E., and K. van Deemter. 1998. On the interpretation of anaphoric noun phrases. *Journal of Semantics* 15(3/4):355–392.
- Kruijff-Korbyová, I., and E. Hajičová. 1997. Topics and centers: a comparison of the salience-based approach and the Centering Theory. *Prague Bulletin of Mathematical Linguistics* 67:25–50.
- Ladd, D.R. 1980. *The Structure of Intonational Meaning: Evidence from English*. Bloomington: Indiana University Press.
- Lakoff, G. 1971. Presupposition and relative well-formedness. In *Semantics: An Interdisciplinary Reader in Philosophy, Linguistics and Psychology*, ed. D. Steinberg and L. Jakobovits. 329–340. Cambridge: Cambridge University Press.
- Lakoff, G. 1987. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. Chicago: The University of Chicago Press.
- Lappin, S., and H. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics* 20(4):535–561.
- Levelt, W. 1989. *Speaking, from Intention to Articulation*. Cambridge: MIT Press.
- Levelt, W.J.M., and S. Kelter. 1982. Surface form and memory in question-answering. *Cognitive Psychology* 14:78–106.
- Levow, G.-A. 1998. Characterizing and recognizing spoken corrections in human-computer dialogue. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL'98)*, 736–742. Montreal, Canada.

- Lewis, D. 1979. Scorekeeping in a language game. *Journal of Philosophical Logic* 8:339–359.
- Maes, A.A. 1991. *Nominal Anaphors and the Coherence of Discourse*. Doctoral dissertation, University of Tilburg.
- Mann, W.C., and S.A. Thompson. 1985. Rhetorical Structure Theory: Toward a functional theory of text organization. *TEXT Journal* 8:243–281.
- McKeown, K., J. Robin, and K. Kukich. 1995. Generating concise natural language summaries. *Information Processing and Management* 31(5):703–733.
- Miller, G. 1995. WordNet: A lexical database for English. *Communications of the ACM* 38(11):39–41.
- Mykowiecka, A. 1991. Natural-language generation – an overview. *International Journal of Man-Machine Studies* 34:497–511.
- Nachtegaal, D. 1997. An evaluation of GoalGetter's accentuation. Report 1142. IPO, Eindhoven, The Netherlands.
- Noord, G. van, G. Bouma, R. Koeling, and M. Nederhof. 1999. Robust grammatical analysis for spoken dialogue systems. *Natural Language Engineering* 5(1):45–93.
- Oberlander, J. 1998. Do the right thing. *Computational Linguistics* 25(3):501–507.
- Odiijk, J. 1995. Generation of coherent monologues. In *CLIN V: Proceedings of the 5th CLIN Meeting*, ed. T. Andernach, M. Moll, and A. Nijholt, 123–131. Enschede, The Netherlands.
- Oviatt, S., M. MacEachern, and G.-A. Levow. 1998. Predicting hyperarticulate speech during human-computer error resolution. *Speech Communication* 24:87–110.
- Pan, S., and K.R. McKeown. 1997. Integrating language generation with speech synthesis in a concept to speech system. In *Proceedings of the Workshop on Concept-to-Speech Generation Systems, ACL/EACL'97*, 23–28. Madrid, Spain.
- Passonneau, R. 1996. Using Centering to relax Gricean informational constraints on discourse anaphoric noun phrases. *Language and Speech* 39(2/3):229–264.
- Pechmann, T. 1989. Incremental speech production and referential overspecification. *Linguistics* 27:98–110.
- Petrič, I., and H. van den Bergh. 1993. Text features affecting listening performance in one way speech communication. In *Analysis and Synthesis of Speech: Strategic Research Towards High-Quality Text-to-Speech Generation*, ed. V. van Heuven and L. Pols. 13–26. Berlin - New York: Mouton de Gruyter.
- Piwek, P. 1998. *Logic, Information & Conversation*. Doctoral dissertation, Eindhoven University of Technology.
- Piwek, P., and E. Krahmer. 2000. Presuppositions in context: Constructing bridges. In *Formal and Linguistic Aspects of Context*, ed. P. Bonzon,

- M. Cavalcanti, and R. Nossun. Applied Logic Series, Vol. 20. Dordrecht: Kluwer Academic Publishers. In press.
- Poesio, M. 1998. Cross-speaker anaphora and dialogue acts. In *Proceedings of the Workshop on Mutual Knowledge, Common Ground and Public Information, ESSLLI'98*. Saarbrücken, Germany.
- Pouteau, X., and L. Arévalo. 1998. Robust spoken dialogue systems for consumer products: A concrete application. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP'98)*, 1231–1234. Sydney, Australia.
- Prevost, S. 1995. *A Semantics of Contrast and Information Structure for Specifying Intonation in Spoken Language Generation*. Doctoral dissertation, University of Pennsylvania.
- Prevost, S. 1996. An information structural approach to spoken language generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, 294–301. Santa Cruz, USA.
- Prince, E.F. 1981. Toward a taxonomy of given/new information. In *Radical Pragmatics*, ed. P. Cole. 223–255. New York: Academic Press.
- Prüst, H. 1992. *On Discourse Structuring, VP Anaphora and Gapping*. Doctoral dissertation, University of Amsterdam.
- Pulman, S. 1997. Higher order unification and the interpretation of focus. *Linguistics and Philosophy* 20:73–115.
- Reiter, E. 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the 7th International Workshop on Natural Language Generation (IWNLG'94)*, 163–170. Kennebunkport, USA.
- Reiter, E. 1995. NLG vs. templates. In *Proceedings of the 5th European Workshop on Natural Language Generation (EWNGL'95)*, 95–106. Leiden, The Netherlands.
- Reiter, E., and R. Dale. 1992. A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING'92)*, 232–238. Nantes, France.
- Reiter, E., and R. Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering* 3(1):57–87.
- Reiter, E., and R. Dale. 2000. *Building applied natural language generation systems*. Cambridge: Cambridge University Press.
- Reiter, E., and C. Mellish. 1993. Optimizing the costs and benefits of natural language generation. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI'93)*, 1164–1169. Chambery, France.
- Rietveld, T., J. Kerkhoff, M.J.W.M. Emons, E.J. Meijer, A.A. Sanderman, and A.M.C. Sluijter. 1997. Evaluation of speech synthesis systems for Dutch in telecommunication applications in GSM and PSTN networks. In *Proceedings of the 5th European Conference on Speech Communication and Technology (Eurospeech'97)*, 577–580. Rhodes, Greece.

- Robin, J. 1994. Automatic generation and revision of natural language report summaries providing historical background. In *Proceedings of the 11th Brazilian Symposium on Artificial Intelligence*. Fortaleza, Brazil.
- Rooth, M. 1985. *Association with Focus*. Doctoral dissertation, University of Massachusetts.
- Rooth, M. 1992. A theory of focus interpretation. *Natural Language Semantics* 1:75–116.
- Rosch, E. 1978. Principles of categorization. In *Cognition and Categorization*, ed. E. Rosch and B. Lloyd. 27–48. Hillsdale: Lawrence Erlbaum.
- Rump, H. H., and R. Collier. 1996. Focus conditions and the prominence of pitch-accented syllables. *Language and Speech* 39(1):1–17.
- Sanderman, A. 1996. *Prosodic Phrasing: Production, Perception, Acceptability and Comprehension*. Doctoral dissertation, Eindhoven University of Technology.
- Schmerling, S.F. 1976. *Aspects of English Sentence Stress*. Austin: University of Texas Press.
- Seneff, S., L. Hirschman, and V. Zue. 1991. Interactive problem solving and dialogue in the ATIS domain. In *Proceedings of the 4th DARPA Speech and Natural Language Workshop*. Pacific Grove, USA. Morgan Kaufmann.
- Sgall, P., E. Hajičová, and J. Panevová. 1986. *The Meaning of the Sentence in its Pragmatic Aspects*. Dordrecht: Reidel.
- Sluijter, A., E. Bosgoed, J. Kerkhoff, E. Meier, T. Rietveld, A. Sanderman, M. Swerts, and J. Terken. 1998. Evaluation of speech synthesis systems for Dutch in telecommunication applications. In *Proceedings of the 3rd ESCA/COCOSDA International Workshop on Speech Synthesis*, 213–218. Jenolan Caves, Australia.
- Soltau, H., and A. Waibel. 1998. On the influence of hyperarticulated speech on recognition performance. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP'98)*, 229–232. Sydney, Australia.
- Sproat, R. 1995. Text interpretation for TtS synthesis. In *Survey of the State of the Art in Human Language Technology*, ed. R. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue. 202–209. Cambridge: Cambridge University Press.
- Steedman, M. 1990. Structure and intonation in spoken language understanding. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL'90)*, 9–17. Pittsburgh, USA.
- Steedman, M. 1996. Representing discourse information for spoken dialogue generation. In *Proceedings of the International Symposium on Spoken Dialogue, ICSLP'96*. Philadelphia, USA.
- Stone, M., and B. Webber. 1998. Textual economy through close coupling of syntax and semantics. In *Proceedings of the 9th International Workshop on Natural Language Generation (IWNLG'98)*, 178–187. Montreal, Canada.

- Strik, H., A. Russel, H. van den Heuvel, L. Boves, and C. Cucchiarini. 1996. Localizing an automatic inquiry system for public transport information. In *Proceedings of the 4th International Conference on Spoken Language Processing (ICSLP'96)*, 853–856. Philadelphia, USA.
- Sturm, J., E. den Os, and L. Boves. 1999. Dialogue management in the Dutch ARISE train time table information system. In *Proceedings of the 6th European Conference on Speech Communication and Technology (Eurospeech'99)*, 1419–1422. Budapest, Hungary.
- Swerts, M., H. Koiso, A. Shimojima, and Y. Katagiri. 1998. On different functions of repetitive utterances. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP'98)*, 483–486. Sydney, Australia.
- Swerts, M., and M. Ostendorf. 1997. Prosodic and lexical indications of discourse structure in human-machine interactions. *Speech Communication* 22:25–41.
- Tanaka, K., K. Hasida, and I. Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL'98)*, 1282–1288. Montreal, Canada.
- Teich, E., and J. Bateman. 1994. Towards the application of text generation in an integrated publication system. In *Proceedings of the 7th International Workshop on Natural Language Generation (IWNLG'94)*. Kennebunkport, USA.
- Terken, J., and J. Hirschberg. 1994. Deaccentuation of words representing 'given' information: effects of persistence of grammatical function and surface position. *Language and Speech* 37(2):125–145.
- Terken, J., and S. Nooteboom. 1987. Opposite effects of accentuation and deaccentuation on verification latencies for given and new information. *Language and Cognitive Processes* 2:145–163.
- Theune, M. 1997a. Contrastive accent in a data-to-speech system. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL/EACL'97)*, 519–521. Madrid, Spain.
- Theune, M. 1997b. GoalGetter: Predicting contrastive accent in data-to-speech generation. In *CLIN VII: Proceedings of the 7th CLIN Meeting*, ed. K. van Deemter, J. Landsbergen, J. Odijk, and G. Veldhuijzen van Zanten, 177–190. Eindhoven, The Netherlands.
- Theune, M. 1999. Parallelism, coherence, and contrastive accent. In *Proceedings of the 6th European Conference on Speech Communication and Technology (Eurospeech'99)*, 555–558. Budapest, Hungary.
- Theune, M. 2000. Language generation in dialogues. In *NWO Priority Programme Language and Speech technology: Final Report*. Chap. 8.

- Theune, M., E. Klabbers, J. Odijk, and J.R. de Pijper. 1997. Computing prosodic properties in a data-to-speech system. In *Proceedings of the Workshop on Concept-to-Speech Generation Systems, ACL/EACL'97*, 39–45. Madrid, Spain.
- Theune, M., E. Klabbers, J. Odijk, J.R. de Pijper, and E. Kraemer. 2000. From data to speech: A generic approach. Submitted to *Natural Language Engineering*.
- Traum, D. 1994. *A Computational Theory of Grounding in Natural Language Conversation*. Doctoral dissertation, University of Rochester.
- Traunmüller, H., and A. Eriksson. 2000. Acoustic effects of variation in vocal effort by men, women, and children. *Journal of the Acoustical Society of America*. To appear.
- Veldhuijzen van Zanten, G. 1998. Adaptive mixed-initiative dialogue management. In *Proceedings of IVTTA 1998*, 65–70. Turin, Italy.
- Veldhuijzen van Zanten, G. 1999a. The interface between the NLP and DM modules in OVIS3. Deliverable 92. Priority Programme on Language and Speech Technology, NWO, The Netherlands.
- Veldhuijzen van Zanten, G. 1999b. The semantics of update expressions. Deliverable 24. Priority Programme on Language and Speech Technology, NWO, The Netherlands.
- Veldhuijzen van Zanten, G. 2000. The update language. In *NWO Priority Programme Language and Speech technology: Final Report*. Chap. 8.
- Verhagen, J-P. 1999. Generating descriptions in context: Implementation and evaluation. Master's thesis, Hogere Informatica, Fontys Hogescholen, Eindhoven, The Netherlands.
- Walker, M. 1998. Centering, anaphora resolution and discourse structure. In *Centering Theory in Discourse*, ed. A. Joshi M. Walker and E. Prince. 401–435. New York: Oxford University Press.
- Walker, M., M. Iida, and S. Cote. 1994. Japanese discourse and the process of Centering. *Computational Linguistics* 20(2):193–232.
- Walker, M., D. Litman, C. Kamm, and A. Abella. 1998. Evaluating spoken dialogue agents with PARADISE: Two case studies. *Computer Speech and Language* 12(3).
- Weegels, M. 1999. Usability evaluation of voice-operated information services: a comparative study of VIOS and ARISE. Deliverable 79. Priority Programme on Language and Speech Technology, NWO, The Netherlands.
- White, M., and T. Caldwell. 1998. A practical, extensible framework for dynamic text generation. In *Proceedings of the 9th International Workshop on Natural Language Generation (IWNLG'98)*, 238–247. Niagara-on-the-Lake, Canada.
- Williams, E. 1997. Blocking and anaphora. *Linguistic Inquiry* 28:577–628.
- Williams, S. 1998. Generating pitch accents in a concept-to-speech system using a knowledge base. In *Proceedings of the 5th International Con-*

- ference on Spoken Language Processing (ICSLP'98)*, 1159–1162. Sydney, Australia.
- Yankelovich, N., G.-A. Levow, and M. Marx. 1995. Designing SpeechActs: Issues in speech user interfaces. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'95)*, 369–376. Denver, USA.
- Young, S., and F. Fallside. 1979. Speech synthesis from concept: A method for speech output from information systems. *Journal of the Acoustical Society of America* 66:685–695.
- Zipf, G.K. 1949. *Human Behavior and the Principle of Least Effort – An Introduction to Human Ecology*. Cambridge: Addison-Wesley Publishing Co.
- Zue, V. 1997. Conversational interfaces: Advances and challenges. In *Proceedings of the 5th European Conference on Speech Communication and Technology (Eurospeech'97)*, 9–18. Rhodes, Greece.
- Zue, V., S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, and L. Hetherington. 2000. Jupiter: A telephone-based conversational interface for weather information. *IEEE Transactions on Speech and Audio Processing* 8(1):85–95.

Samenvatting

Taalgeneratie is de automatische productie van een tekst op basis van niet-talige informatie, die bijvoorbeeld afkomstig kan zijn uit een database. Dit proefschrift richt zich in het bijzonder op de generatie van teksten die bedoeld zijn om in gesproken vorm te worden gepresenteerd. Omdat de gegenereerde teksten moeten worden uitgesproken, is de uitvoer van het hier besproken taalgeneratie-systeem verrijkt met tekens die de prosodische eigenschappen van de tekst weergeven. Het gaat hierbij in het bijzonder om tekens die de plaats aangeven van accenten en frase-grenzen.

Zoals de titel van het proefschrift al aangeeft, richt het hier gepresenteerde onderzoek zich op taalgeneratie *in context*. Het woord “context” wordt hier gebruikt in drie verschillende betekenissen.

1. Centraal in dit proefschrift staat de *taalkundige context* van te genereren uitingen, dat wil zeggen, datgene wat reeds gezegd is. Onderzocht is hoe informatie over de taalkundige context kan worden gebruikt om samenhang en natuurlijkheid van de gegenereerde teksten te bevorderen. Daarbij is in het bijzonder gekeken naar het effect van de taalkundige context op de generatie van verwijzende uitdrukkingen en de plaatsing van accenten.
2. Daarnaast wordt aandacht besteed aan taalgeneratie in de context van twee verschillende interactie-vormen: *monoloog* en *dialog*. Bij taalgeneratie in de vorm van een monoloog bestaat de taalkundige context van een uiting uit de tekst die het systeem tot dusver heeft gegenereerd. Bij taalgeneratie in een dialoog moeten echter ook de uitingen van de gebruiker tot de taalkundige context gerekend worden, maar het is niet altijd zeker of het systeem deze uitingen goed heeft herkend. Dit kan (onder andere) van invloed zijn

op accentuering en op de generatie van verwijzende uitdrukkingen.

3. Tenslotte richt dit proefschrift zich op taalgeneratie in de context van een *data-naar-spraak* systeem, waarin taal- en spraakgeneratie gecombineerd worden. Het meer specifieke kader wordt gevormd door D2S, een generiek data-naar-spraak systeem dat op het IPO is ontwikkeld. Het hier beschreven onderzoek heeft geleid tot diverse verbeteringen van de taalgeneratie-module in D2S.

Hoofdstuk 1 van dit proefschrift geeft een beknopt overzicht van taal- en spraakgeneratie. De basis-taken van een taalgeneratie-systeem worden besproken, en enkele veelgebruikte methoden voor spraakgeneratie worden kort uiteengezet. Ook wordt een overzicht gegeven van taalkundige informatie die relevant is voor de prosodie van de gegenereerde tekst. In een data-naar-spraak systeem hoeft deze informatie niet achterhaald te worden door middel van tekstanalyse, maar kan deze direct geleverd worden door de taalgeneratie-module.

In Hoofdstuk 2 wordt de werking van D2S uitgelegd aan de hand van voorbeelden uit het GoalGetter-systeem, dat binnen dit onderzoek is ontwikkeld. De nadruk ligt daarbij op de taalgeneratie-module van D2S, die in detail beschreven wordt. Het belangrijkste kenmerk van D2S is dat taalgeneratie wordt gedaan door middel van zogenaamde 'syntactische templatens': syntactische boomstructuren met open posities waarin variabele informatie kan worden ingevuld. Door het gebruik van deze syntactische templatens, die domein-specifiek zijn en met de hand worden gemaakt, wordt de taak van zins-realiseren in D2S aanzienlijk vereenvoudigd. Het accent ligt in D2S op andere generatie-taken, zoals de generatie van verwijzende uitdrukkingen en bepaling van de prosodie. Daarbij wordt gebruik gemaakt van een representatie van de taalkundige context, die onder andere weergeeft welke entiteiten genoemd zijn en hoe en wanneer er naar werd verwezen. Deze informatie wordt onder meer gebruikt om na te gaan welke antecedenten beschikbaar zijn voor anaforische verwijzende uitdrukkingen, en om de plaatsing van accenten te bepalen: woorden die verwijzen naar reeds genoemde informatie krijgen in de regel geen accent.

In Hoofdstuk 3 wordt uiteengezet hoe informatie over de taalkundige context gebruikt kan worden om te bepalen welke woorden in de gegenereerde tekst contrastieve informatie uitdrukken. Dit is belangrijk om een natuurlijke prosodie te verkrijgen, aangezien contrastieve informatie altijd geaccentueerd dient te worden, zelfs als deze informatie al eerder genoemd is. In de meeste bestaande benaderingen wordt contrast gerelateerd aan een vorm van parallelisme tussen opeenvolgende zinnen.

Om dit verband te verifiëren is een klein perceptie-experiment uitgevoerd, waarvan de resultaten erop wijzen dat mensen contrast-accent gebruiken om de nadruk te leggen op de punten van verschil tussen gelijksoortige gebeurtenissen. (Een voorbeeld is: *De PAUS kuste de GROND; het STAATSHOOFD kuste HEM.*) Om dit te modelleren in een taalgeneratie-systeem is het nodig om niet alleen bij te houden welke entiteiten genoemd zijn in de taalkundige context, maar ook bij welke gebeurtenissen deze entiteiten betrokken waren en welke rol zij daarin speelden. Gebaseerd op dit idee is een praktische methode ontwikkeld om contrast te bepalen, gebaseerd op de datastructuren die in D2S voor taalgeneratie worden gebruikt. Als twee opeenvolgende zinnen hetzelfde type datastructuur uitdrukken, betekent dit dat zij dezelfde soort gebeurtenis beschrijven, en daarom worden zij als potentieel contrastief beschouwd. Een accent wordt toegekend aan de frasen die de punten van verschil tussen de datastructuren uitdrukken. Een van de voordelen van de voorgestelde aanpak is dat die het mogelijk maakt om automatisch het contrast te herkennen tussen zinnen die weliswaar dezelfde soort gebeurtenis uitdrukken, maar daarvoor verschillende oppervlaktestructuren gebruiken.

Hoofdstuk 4 richt zich op een andere belangrijke taalgeneratie-taak: de generatie van verwijzende uitdrukkingen, waarbij gekozen moet worden welke eigenschappen van een entiteit worden uitgedrukt. Deze keus hangt niet alleen af van de eigenschappen van de andere entiteiten in het domein, waarvan de beoogde referent moet worden onderscheiden, maar ook van de prominentie van de referent. Deze prominentie hangt op zijn beurt af van, onder andere, de taalkundige context: een entiteit waarnaar onlangs is verwezen, is in de regel prominenter dan een entiteit die niet is genoemd. Deze gedachte vormt het uitgangspunt voor de aanpassing van een welbekend algoritme voor de generatie van definiëte beschrijvingen. Het aangepaste algoritme houdt rekening met de taalkundige context bij het kiezen van de eigenschappen die in een beschrijving worden opgenomen. Een eigenschap wordt alleen geselecteerd, als die meehelpt om de bedoelde referent te onderscheiden van andere entiteiten die minstens zo prominent zijn. Op deze manier wordt het mogelijk om gereduceerde anaforische verwijzingen te produceren. In het algemeen wordt een volledige beschrijving (bijvoorbeeld *de grote zwarte hond*) alleen gebruikt wanneer de betreffende entiteit voor het eerst genoemd wordt. Naargelang de entiteit prominenter wordt in vergelijking met de andere entiteiten in het domein, zijn er minder eigenschappen nodig om hem te beschrijven (bijvoorbeeld alleen *de hond*). Doordat de volledige beschrijvingen niet steeds worden herhaald, worden de samenhang en natuurlijkheid van de gegenereerde tekst bevorderd. Om prominentie te

modelleren wordt een methode voorgesteld die een synthese is van twee bestaande methoden. Daarnaast is een experiment uitgevoerd dat aantoonst dat mensen inderdaad meestal de voorkeur geven aan gereduceerde anaforische beschrijvingen. Tenslotte is het algoritme verder aangepast om ook andere soorten verwijzende uitdrukkingen te genereren, zoals persoonlijke voornaamwoorden en relationele beschrijvingen.

Hoofdstuk 5 onderzoekt de verschillen tussen taalgeneratie in een monoloog en een dialoog. Dit hoofdstuk beschrijft onder andere hoe de taalgeneratie-methode van D2S is aangepast voor gebruik in een dialoog-systeem, en bespreekt de gevolgen van een dialoog-context voor accentuering en de generatie van verwijzende uitdrukkingen. In een dialoog tussen mens en machine is het, bijvoorbeeld door problemen met automatische spraakherkenning, niet altijd voor beide partijen duidelijk wat zich precies in de taalkundige context bevindt. In zo'n geval lijken de accentuering of anaforische verwijzing onmogelijk te zijn. Daarnaast kan het systeem geen informatie verstrekken op basis van gegevens die door de gebruiker zijn aangedragen, voordat mogelijke misverstanden zijn opgehelderd. Zoals gezegd bevat de taalkundige context in een dialoog zowel uitingen van het systeem als van de gebruiker, en deze laatste kunnen belangrijke informatie leveren over de eventuele aanwezigheid van communicatiestoornissen. De analyse van een corpus met mens-machine dialogen wijst uit dat de uitingen waarmee gebruikers reageren op communicatiestoornissen, gemarkeerde taalkundige en prosodische kenmerken hebben. Deze kenmerken kunnen het systeem helpen om communicatiestoornissen automatisch te detecteren.

Het in dit proefschrift beschreven onderzoek toont aan dat de uitvoering van diverse taalgeneratie-taken verbeterd kan worden door beter rekening te houden met de context. Of andere generatie-taken evenveel baat kunnen hebben bij ruimere aandacht voor contextuele aspecten is een interessant onderwerp voor verder onderzoek.

Curriculum Vitae

Mariët Theune was born in Vlissingen, on July 19th 1971. From 1983 to 1989, she attended the Rijksscholengemeenschap Scheldemond in Vlissingen, where she obtained her VWO diploma. From 1989 to 1995, she studied Algemene Letteren ('general arts') at Utrecht University, specialising in the syntax and semantics of natural language. As part of her study, from September 1993 to March 1994 she carried out a practical training in computational linguistics at the University of Amsterdam, in an EC project aimed at specifying a linguistic model of discourse. In the period that followed, she combined working in a bookstore with writing her Master's thesis, which dealt with lexical semantics. She graduated (cum laude) in August 1995. From 1996 to 2000, she was employed as a PhD student by NWO, the Netherlands Organisation for Scientific Research, within the framework of the Priority Programme Language and Speech Technology (TST). During this period, she was stationed at IPO, Center for User-System Interaction at the Eindhoven University of Technology.